The Integration of Multiple and Diverse Knowledge Representation Paradigms using a Blackboard Architecture

Thesis

UNRESTRICTED

The Integration of Multiple and Diverse Knowledge Representation Paradigms using a Blackboard Architecture

A Thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy in Computer Science

Alan Harrison BA MSc

Faculty of Mathematics and Computing
The Open University

December 1994

Volume 1 of 3

Date of submission: 20 December 1994
Date of award: 22 June 1995

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

There is increasing evidence that designers of future real-time embedded systems are turning to knowledge-based techniques in order to solve complex problems where algorithmic techniques have failed to produce a solution. In addition, many applications have been mandated to use the Ada programming language for all implementation software, including the knowledge-based components.

This thesis identifies three essential requirements needed to support the construction of these systems: first, the need to provide a library of Ada knowledge-based components that supports a variety of knowledge representation paradigms to model the diverse expert domains being encountered in complex applications; second, the need to provide the user with the means of creating and controlling multiple independent instances of the knowledge-based components to cope with the complexity and scale of the implementations; and third, the need to provide an integrating architecture in which the knowledge-based components may be embedded directly into an application environment.

These requirements have been satisfied by using ideas derived from the concept of abstract data types to construct a library of knowledge-based components; the components have been called abstract knowledge types. Subsequently, multiple instances of the abstract knowledge types have been integrated in modules called knowledge sources, which model specific problem knowledge domains. The knowledge sources have been used to construct a blackboard architecture.

The abstract knowledge types have been used to build a prototype university timetabling system in order to demonstrate their use. The research has shown that the abstract knowledge type integration approach results in a uniform implementation strategy for both conventional and knowledge-based components.

Contents

Chapter 6
Implementation of a
Frame Abstract Knowledge Type

Chapter 7
Implementation of a
Blackboard Abstract Knowledge Type

List of Figures

List of Tables

List of Charts

Acknowledgements

I would like to record my thanks to Dr Pete Thomas for his excellent supervision and guidance through out this research.

In addition, I would also like to thank Dr John Miles for the loan of his M3 data fusion model, as it was while working on the translation of the model from Ada into several other languages that the idea for this research germinated.

Finally, my deepest thanks go to my wife Andrea for the unfailing support she has given me over many years of study.

Publications

The concept of Abstract Knowledge Types was presented at the Institute of Electrical Engineers Colloquium on "Real-time Knowledge-based Systems" in a paper "Knowledge Representation in Real-time Knowledge-based Systems", Harrison,A. & Thomas,P., November 1992.

Chapter 1

Introduction

1.1 Background

There is increasing evidence that designers of future real-time embedded systems are turning to knowledge-based techniques in order to solve complex problems where algorithmic techniques have failed to produce a solution. Examples can be found in the proposed designs for future space exploration vehicles and military command and control systems.

In the latter part of the 1990s an international effort will launch Space Station Freedom, Woods [149], into low Earth orbit which will enable research into materials science, the physiological effects of micro-gravity on humans and to serve as a transfer station for exploration of the solar system. The on-board data management system will operate across distributed real-time processors and will support the use of knowledge-based systems with the intention of enhancing the capabilities of the Freedom Station.

Here on Earth, Naval researchers, Brander and Miles [21, 106], are currently testing a technical demonstrator which aims to show the suitability of using knowledge-based techniques to assist commanders make the correct decisions when confronted with overwhelming amounts of data. For example, data is received from multiple primary and secondary radars, sonars, electronic surveillance sensors

and data links from other surface, sub-surface and airborne platforms. The demonstrator uses knowledge-based techniques to fuse the data and produce a synthetic tactical picture of the battle arena. In the near future, the functionality of the system will be extended to address the problems of situation assessment and resource allocation; situation assessment uses information derived from the tactical picture to predict the most probable meaning of the fused data, and resource allocation determines the most appropriate response to the anticipated threat detected during the assessment phase.

1.2 Perceived Problems

The designers of complex applications such as these must address a number of common problems before viable solutions can be implemented.

1.2.1 Diversity and Multiplicity of Knowledge

In the design of Freedom, Woods [149] identifies the need to provide knowledge-based assistance to support enhanced diagnostic and predictive maintenance of complex electronic systems, together with the maintenance of thermal equilibrium between Freedom Station components. In addition, it is planned to use knowledge-based techniques in subsystems such as data management, guidance navigation, crew health care and software scheduling.

In command and control, Brander [21] identifies the need for expertise in Naval plans, Standard Operating Procedures and the use of geographic, oceanographic and meteorological knowledge, together with knowledge regarding mission objectives, allocated resources, and intelligence in order to provide the basis of a modern automated command and control system. Furthermore, although the research into

multi-sensor data fusion described by Miles [106], the initial process in the proposed automated command and control system, has concentrated on the application of a single knowledge-based technology, coupled with the use of database data, to establish a tactical picture of the Naval operational arena, it is quite probable that data fusion could also benefit from the use of a variety of knowledge-based paradigms to model this complex problem.

These two examples are sufficient to suggest that implementors of such systems will need to:

- Use diverse knowledge representation paradigms and associated inference strategies to model the diversity of knowledge types that exist in the application domains; diversity is implicit in the requirement for diagnosis, prediction, management, control, navigation, scheduling, health care, situation assessment and resource allocation. It is unlikely that one knowledge representation paradigm can model this range of knowledge types and associated inference strategies. Baum [10] also recognises this need for a variety of representations and states that it "..is advantageous, since having a good mapping from the problem domain to the implementation increases the clarity of the application." Furthermore, Craig [43] states that "The problems being tackled by knowledge-based systems are becoming increasingly complex. This complexity is reflected in the diversity of knowledge required to solve problems and the difficulty in finding adequate solutions".

- Replicate the knowledge representation paradigms so that multiple independent instances may be distributed throughout the implementations; more than one instance of a particular paradigm will

be needed to cope with the complexity and scale of the anticipated applications.

- Embed the multiple and diverse knowledge representation paradigm instances in the context of other conventional software engineered components. For example, Naedel [111] anticipates that 90% of a command and control system will be built using conventional components which have been implemented using procedural languages, while only 10% of the implementation will use knowledge-based techniques. Moreover, Partridge [119] states that "..we shall have to incorporate AI[1] into practical software systems" in order to meet the demand for more software power; a requirement that is implicit in the space and military examples.

- Use multiple co-operating knowledge representation paradigm instances in order to solve complex issues; the situation assessment and resource allocation tasks described by Brander [21] imply the existence of multiple co-operating experts. For example, situation assessment will require surface, sub-surface and air defence expertise in order to establish the meaning of the information presented by the tactical picture. Combinations of surface, sub-surface and airborne platforms need to be grouped and the threat posed by the groupings evaluated. Furthermore, resource allocation requires the knowledge provided by experts in surface, sub-surface and airborne tactics to establish the most effective response to the perceived threat.

---

1 AI - Artificial Intelligence. Partridge states that "In Europe particularly, the term Knowledge-Based System is taken as virtually synonymous with AI systems".

1.2.2 Controlling Diversity and Multiplicity

The anticipated diversity and multiplicity of the knowledge-based instances that will be needed to solve such complex application problems suggest the need for an integration and control strategy.

1.2.3 Implementation Language and the Ada Mandate

One of the issues facing the designers of real-time knowledge-based systems is that of choosing an appropriate implementation language. Ideally, such a language should provide real-time performance and support software engineering principles, while at the same time provide a wide selection of knowledge representation paradigms to match the diversity of knowledge types that will be found in complex application domains. Unfortunately, there is no real-time language that supports the wide variety of knowledge representations needed to model these problems.

Existing languages, such as LISP, which are popular for implementing knowledge-based systems, only provide limited knowledge representation primitives, and do not offer the range of high-level representations that will be needed. In addition, Naedal [111], Diaz-Herru [48] and Williams [146] point out that these languages are not usually associated with applications which require the speed, reliability, integrity and maintainability normally expected of real-time operation.

Both application examples given above are implementing their knowledge-based solutions in Ada, a procedural language designed for implementing real-time embedded systems. Moreover, Collard [32], Hintz [76] and Sibley [134], show that system designers, particularly in military applications, are mandated to use Ada as their implementation language. Since Ada does not provide the knowledge

representations and associated inference operations needed to implement the knowledge-based solutions, work needs to be done to establish how the gap between language and knowledge-based component implementation may be bridged.

1.3 Research Aims

This thesis sets out to show that a variety of independent, generally applicable knowledge representation paradigms can be implemented using the real-time programming language Ada and integrated using a generally applicable control architecture. In particular, the aims were to:

- Provide a library of independent software components, to support a variety of knowledge representation paradigms, that can be used to model the diverse expert domains being encountered by the designers of future complex real-time systems. The components should be easy to integrate into different applications, and provide a means of prototyping knowledge-based solutions directly in environments which are dominated by conventional components that have been implemented using procedural languages and software engineering principles.

- Provide the user of the library with a means of creating multiple independent instances of the knowledge-based components, and the means of controlling the instances in order that designers may model problems requiring consultation between multiple co-operating experts.

- Provide a means of integrating and controlling the anticipated complexity of the assembled diverse and possibly multiple knowledge-based components.

- Build a prototype university timetabling system to demonstrate the use of the knowledge-based components and control architecture.

1.4 Thesis Structure

Chapter 2 lays down the background theory regarding a selection of knowledge representation paradigms that were used in the prototyping experiments, and discusses the Ada features that have been used to implement the knowledge-based components; the components have been called abstract knowledge types because of their similarity to abstract data types. Previous work in this area is also introduced.

Chapter 3 reviews several approaches to the integration of diverse knowledge representation paradigms. In particular, it describes hybrid knowledge representation tools, but discards these in favour of a general problem solving blackboard architecture which can be used as an embedded component to integrate abstract knowledge type instances.

Chapters 4-6 describe the implementation and testing of the abstract knowledge type components; these are logic, rules and frames. The chapters show how the components have been implemented using the Ada generic construct which enables multiple independent instances to be instantiated. In addition, the means by which a logic abstract knowledge type instance communicates with other logic instances is described. The results of each experiment were derived using dynamic analysis and show that dynamic string manipulations consume a significant amount of processor time.

Chapter 7 outlines the design and implementation of a generic blackboard space which can be instantiated to meet the requirements of an application. The chapter shows how a hierarchy of abstract data types and abstract knowledge types can be generated to form the framework of a blackboard architecture.

Chapter 8 describes an experiment which uses multiple instances of the abstract knowledge types to demonstrate the integration strategy. In addition, the use of co-operating abstract knowledge type instances is demonstrated. The experiment was based on the requirement for an automated university timetabling system. From inputs of module codes the system uses rule based abstract knowledge type instances to construct the timetables. Logic abstract knowledge type instances have been used to allocate period and room resources, while frame abstract knowledge type instances have been used to store staff, room and period details. The results of the experiment confirmed the significant use of processor time by dynamic string operations.

Chapter 9 discusses the results of the experiments described in Chapters 4-8 and details the conclusions that have been drawn. The most important conclusion is that the use of abstract knowledge types encourages a uniform, software engineered implementation approach to be applied to both conventional and knowledge-based components; all components are implemented and manipulated in the same way. In addition, the existence of a library of abstract knowledge type components provides the basis for prototyping knowledge-based solutions in the application implementation environment The chapter concludes with suggestions for further work.

Chapter 2

Knowledge Representation Paradigms

2.1 Introduction

Chapter 1 identified future applications where system designers are planning to use knowledge-based techniques to solve complex problems in real-time embedded systems. In each case the domain knowledge is diverse. Consequently, in order to provide the best model of an application knowledge domain, designers will have to use a diversity of knowledge representation paradigms. The aim of this chapter is to describe the knowledge representation paradigms chosen to test the thesis defined in Chapter 1 and to review the knowledge representations already implemented in Ada.

2.2 Knowledge Representation

Wirth [148] describes a computer program as "Data + Algorithm = Program". In a similar way, knowledge-based systems are described as "Knowledge + Inference = Knowledge-based System".

In the algorithmic approach, data is used to represent values from the problem domain upon which the algorithm operates. Data is usually implemented as instances of primitive types, such as integer, character, float or string. In addition, the data values may be grouped in data structures such as arrays or records. Furthermore, a designer may define abstract data types such as stacks and

queues, which are generally applicable across a variety of applications. The relationships between data items are implicitly defined in the algorithms by appropriate control constructs, together with other control constructs which are needed to solve an application problem. Consequently, a change in a data definition or a control construct requires a re-compilation of the algorithm.

In the knowledge-based systems approach, the definition of knowledge has a broader scope than that for data in the algorithmic approach, but will include data, a set of facts known about the problem domain, as part of the knowledge definition. In addition, the knowledge will include an explicit definition of the relationships that exists between the facts. The set of relations permit new knowledge, knowledge which is implied by the explicit fact definitions, to be extracted by an inference process. Unlike the algorithmic approach, where the data definitions, relationships and control constructs are encapsulated in the algorithm, knowledge in a knowledge-based system is separated from the control inference mechanism. Consequently, a change in the knowledge base definition does not require a re-compilation of the inference mechanism. Furthermore, the inference mechanism is usually application independent. These characteristics are important in applications where the knowledge definition needs to be changed as events evolve. For example, the rules of engagement encapsulated in a command and control system may need to be changed as a battle develops; this would be difficult in an implementation where re-compilation of the system was necessary.

Bench-Capon [12] defines knowledge representation as "a set of syntactic and semantic conventions that makes it possible to describe things". Brachman [19, 20] identifies the role of knowledge representation as "How do we impart knowledge of the world to a robot or other computational system so that, given an appropriate reasoning capacity, that knowledge can be used to allow the system to adapt to and

exploit its environment". In practice, a computer-based knowledge representation models 'things' in a real world problem domain. These knowledge representations have at least two forms; first, an external form, usually a file containing a textual description of the knowledge written using a pre-defined syntax and semantics; second, an internal form, which models the external representation, together with associated inference operations to implement the semantics.

Research has produced many different knowledge representations, some of these are specific to a given problem, whereas others are generally applicable. The most common, generally applicable, knowledge representations described in the literature are logic, rules, semantic networks and frames. Others include objects and neural network representations and representations for control and time.

However, the focus of this research is on the integration of knowledge representation paradigms rather than the advancement of knowledge in the topic itself. Consequently, the rest of this chapter describes the three knowledge representations chosen to form the basis of an integration experiment. The three representations were chosen because of their general applicability to a wide range of applications; these are logic, rules and frames.

2.3 Logic Knowledge Representation

Since the time of Aristotie(384-322BC), philosophers have used logic and inference rules to represent human knowledge and human reasoning. Moreover, numerous authors use logic as the basis for introducing the concept of knowledge representation in knowledge-based systems. In particular, Nilsson [118], Rich [125], Ringland & Duce [126], Bench-Capon [12] and Lucas & Van Der Gang [102] introduce the idea of knowledge representation through propositional logic.

2.3.1 Propositional Logic

Propositional logic is a formal language with a well defined syntax and semantics that uses propositions to represent knowledge. A proposition is perceived to be either true or false. Compound propositions, which are also perceived to be true or false, can be formed by joining propositions with logical connectors. However, since the truth values in propositional logic apply only to whole propositions, whether simple or compound, the language is unable to represent the state of the component parts of a proposition. Furthermore, propositions are unable to generalise about situations which are similar, but may involve different object instances. Consequently, the expressive power of propositional logic is limited.

2.3.2 First-order Predicate Logic

On the other hand, first-order predicate logic is able to represent and reason about the component parts of a proposition. Moreover, a simple, but very powerful inference strategy has been developed for predicate logic that can be automated efficiently. In particular, the programming language PROLOG is based on the principles of first-order predicate logic and an inference strategy known as resolution.

In first-order predicate logic a proposition is represented, in the simplest case, by a formula comprising a predicate followed by an ordered sequence of arguments, where the predicate represents a relationship between the arguments. In addition, first-order predicate logic uses quantifiers, the universal quantifier $\forall$ and the existential quantifier $\exists$, to enable the use of variables within a logic formula. Compound formula may be formed by joining predicates with logical connectors. Such compound formula may be complex, and there may be many different formula

that can represent the same logical information. Consequently, automatic manipulation of predicate logic formula, by computer, is difficult. Therefore, in order to automate the manipulation of logic formula efficiently, predicate logic notation is transformed into a normal form. A well established set of logical transformations exist that can be used to turn a first-order predicate logic formula into a normal form. Moreover, Nilsson [118], Clocksin & Mellish [30], Rich [125] and Lucas & Van Der Gaag [102] present general algorithms to achieve this task. In the disjunctive normal form, a clause is represented by a disjunction of conjunctions, for example $(A \cap B \cap C) \cup (D \cap E \cap F)$, but in the conjunctive normal form a clause is represented by a conjunction of disjunctions, for example $(A \cup B \cup C) \cap (D \cup E \cup F)$. However, in the clausal form, a clause is represented by an implication with positive literals on the left and negative literals on the right, for example, $(A \cup B \cup C) \leftarrow (D \cap E \cap F)$. Although these forms are logically equivalent to the first-order predicate logic from which they are derived they are more computationally efficient, however, they are less expressive.

2.3.3 Resolution and Unification

The inference strategy used to automate the proof process was developed by Robinson [127] and is called resolution. In particular, resolution by refutation is commonly used; this can be applied to all the normal forms. In this strategy, the negation of a postulated goal is added to a set of clauses derived by the process outlined in 2.3.2. Resolution is then achieved by searching the set of clauses for two particular clauses, called parent clauses, in which one clause contains a negation of a literal contained in the other clause. The resolution principle then allows a new clause to be formed, the resolvent clause, which is the disjunction of the parent clauses less the two complementary literals. Should the resolvent be the empty clause, then a contradiction is assumed and the original negated goal proved

false. On the other hand, if the resolvent is not the empty clause, then the resolvent is added to the set of clauses and the resolution process repeated. This process continues until a contradiction is found, or until no progress can be made.

During the process of resolution, when a search is being made for two parent clauses to resolve, the presence of variables makes the task of recognising complementary literals more difficult than would be the case if all the components of a clause were constants. To overcome this difficulty, substitutions are made for variables so that literals are matched and subsequently resolved; this process is known as unification and forms a major part of the proof process.

2.3.4 PROLOG

PROLOG, Clocksin & Mellish [30], Bratco [22] and Kluzniak & Szpakowicz [90], is a declarative programming paradigm which was developed in the early 1970s as the result of experiments in using logic as a programming language. The declarative paradigm makes explicit the logical relationships within a problem, but keeps implicit the control mechanisms which are used to solve the problem. It is this declarative characteristic of PROLOG which makes it an attractive knowledge representation paradigm.

2.3.4.1 PROLOG Knowledge Base

Clocksin & Mellish [30] show that PROLOG is based on a restricted first-order predicate logic clausal form, which limits the left hand side of a clause to only one predicate; this form is called a Horn clause. A PROLOG knowledge base is simply a collection of Horn clauses. Each clause takes one of two forms: a fact, which is a Horn clause with just a left hand side, or a rule. An example of a small PROLOG

knowledge base extracted from Bratco [22] is shown in Fig 2.1 which represents the classic monkey and banana problem; constants start with a lowercase letter while a variable starts with an uppercase letter or a '_'.

```
move( state( middle, onbox, middle, hasnot),
      grasp,
      state( middle, onbox, middle, has)).
move( state( P, onfloor, P, H),
      climb,
      state( P, onbox, P, H)).
move( state( P1, onfloor, P1, H),
      push( P1, P2),
      state( P2, onfloor, P2, H)).
move( state( P1, onfloor, B, H),
      walk( P1, P2),
      state( P2, onfloor, B, H)).
canget( state( _, _, _, has)).
canget( S1) :- move( S1, M, S2), canget( S2).
```

Fig 2.1 A PROLOG knowledge base.

Note that the four move facts represent the valid moves that the monkey can make to acquire the banana; each clause defines the start state, type of move and the resultant state. The rule canget is a recursive definition of how the monkey can get the banana; the fact canget defines the terminating condition for the recursive rule. The knowledge base is queried by specifying a goal such as canget(state(atdoor, onfloor, atwindow, hasnot)). This is explored further in Chapter 4.

2.3.4.2 PROLOG Inference Strategy

The inferencing strategy in PROLOG is based on unification and resolution, which is implemented using pattern matching, backward chaining and backtracking. Given an initial goal, the inference engine searches the left hand side of each of the PROLOG clauses for a match. On finding a match, if the matched clause is a fact, and unification is possible, then the goal is satisfied. If the matched clause is a rule, and

unification is possible, then the initial goal is replaced by the subgoals on the right side of the matching rule. The inference engine then backward chains on each of the subgoals trying to satisfy each in turn. This process continues until all subgoals have been satisfied, thus satisfying the initial goal, or until the knowledge base is exhausted , so failing the initial goal. Should intermediate subgoals fail during this process, then the inference engine backtracks to a previous subgoal and tries to re-satisfy; this strategy is particularly useful for implementing a deductive data base.

2.4 Rule Knowledge Representation

Rules are a popular form of knowledge representation that have been used in many diverse applications from medical diagnosis to data fusion. In addition, many commercial expert systems shells use rules as the primary knowledge representation paradigm.

A rule is the encapsulation of a relationship between fragments of knowledge. There is no standard syntax for writing rules, but the form of a rule is similar in most implementations. Each rule has two main parts; a condition, often called the antecedent, and an action, often called the consequent. The condition part, when compared with what is known about the problem domain at any instance, must evaluate to true before the action part can be activated. The process of comparison and subsequent activation of the rule action is performed by an inference engine.

From this description it can be seen that a computer implementation of the rule paradigm must provide a means of storing what is known about a problem domain, a representation of the rule base and an algorithm to perform the inference process.

2.4.1 Fact base

The detail known about a problem domain at any instance is recorded in a fact base. Like the rules, there is no standard syntax or way of representing the facts known about a problem; it will be dependent on the nature of the domain knowledge. The fact base may be an integral part of the rule-based system or may be shared between independent rule-based components as will be shown in Chapter 8.

2.4.2 Rule Base

Unlike PROLOG, where facts and rules are mixed together in the knowledge base, in a rule-based system the facts and rules are normally kept separate. Again there is no standard way of implementing the rule structure; this will depend on the implementation language. However, since the number of antecedents and consequents may vary from rule to rule, the implementation structure should be flexible. An example of a rule used in the data fusion process implemented by Miles [106] is shown in Fig 2.2.

```
if    there is a radar track(rt)
and   any muiti-radar track(mrt)
and   position and velocity differences of rt and mrt meet criteria
then  create a tentative correlation between rt and mrt
```

Fig 2.2 A data fusion rule.

2.4.3 Rule Base Inference Strategies

There are two common inference strategies used in rule-based systems; these are forward and backward chaining. The choice of strategy depends on the application.

2.4.3.1 Forward Chaining

In the forward chaining strategy, the detail recorded in the fact base is compared with the antecedents of each rule in the rule base. If each of the antecedents in a rule match some entry in the fact base, then the selected rule is said to be ready to fire; that is the action can be activated. The rule base is searched from the first rule to the last rule, in sequence, identifying all the rules that are ready to fire, based on the current content of the fact base. The identified rules are placed on a list called the agenda. When the search is complete the rules on the agenda are fired. The action of firing a rule will normally change the content of the fact base. A change in the fact base triggers another search on the rule base and subsequent additions to the agenda. This cyclic process of match and fire continues until there are no more rules to fire. The result of the process can be viewed as a chain of fired rules which develops by proceeding from the antecedent to consequent of each rule in turn; a left to right or forward direction. Consequently, the process is known as forward chaining or data driven inference. This inferencing strategy is ideal for solving problems which require prognosis. For example, Miles [106] uses this strategy to fuse the output from electronic sensors to form a tactical picture of a Naval battle scenario.

At the point a rule is selected to fire there is often a choice between which rule to use; the choice is known as conflict resolution. Various conflict resolution strategies are used, from a sequential selection to selecting the rule with most antecedents. In addition, rules are often partitioned to make the search more efficient and meta rules used to guide the inference engine as to which partition to search.

2.4.3.2 Backward Chaining

In the backward chaining inference strategy, the aim is to prove that a goal can be derived from the set of facts that have been established in the fact base. Consequently, a postulated goal is compared with the consequent of each rule in the rule base. When a goal matches a consequent, the current goal can be proved if the antecedents of the matched rule are present in the fact base or can be proved in turn using the rules in the rule base. In the case where the antecedents match existing entries in the fact base, the goal is proved and the backward chaining stops, or the system can backtrack and try to establish other ways of proving the goal. In this way, all possible ways of proving an initial goal can be established. In the case where the antecedents are not currently established in the fact base, the current goal is replaced by the antecedents of the matched rule, which become subgoals. The process then proceeds to establish that these new subgoals can be proved from the current state of the knowledge base by comparing each subgoal, in turn, with the consequents of the rules in the rule base. The process stops when all subgoals have been proved, showing that the original goal can be proved from the original state of the knowledge base. Should any subgoal fail to be satisfied, then the inference engine backtracks to find alternate evidence to support the subgoal. If, at the end of all possible search paths through the rule base, there is an unsatisfied subgoal, it is concluded that the original goal can not be proved from the original state of the knowledge base. The result of the process can be viewed as a chain of rules which develops by proceeding from the consequent to antecedents of each rule in turn; a right to left or backward direction. Consequently, the process is known as backward chaining or goal driven inference. This inferencing strategy is ideal for solving problems which require diagnosis. For example, modern electronic systems diagnostic tools use backward chaining to establish component faults.

2.5 Frame Knowledge Representation

There are many examples from the natural world, and from the world of man-made systems, that form taxonomies and structured architectures. For example, a biological classification system or the module architecture in a command and control system. The knowledge associated with this type of problem is best represented by a paradigm which is able to model the structural features of the problem domain. One of the most common structured knowledge representation paradigms is frame representation. The concept of frames was originally used by Minsky [109] to represent computer vision.

2.5.1 Frame Structure

One of the main problems when trying to implement a frame-base system is that there is no agreed definition in the literature of what the structure should be. However, frame implementations have common characteristics that are described by various authors. A frame is perceived to be a clustering of knowledge about the attributes of an entity or object which exists in the problem domain. A frame can describe the general properties of a class of objects, where the detailed values of the class attributes are not specified, but where default values can be given. Alternatively, a frame can represent a specific instance from the problem domain which encapsulates the particular values associated with the particular object being represented. The frames comprise slots which are filled with the required knowledge. The fillers can be simple values, other frames, procedures which can be activated automatically by reference to a slot, and rules which can be used to influence the inference process. In addition, the slots can have a number of facets which in turn store more detailed knowledge.

The definition of a class may be extended by the addition of a sub-class; the sub-class inherits the slots of its superclass. A class frame is a generalisation of the instance frames. The class frames and instance frames are linked to form a hierarchical structure such as that shown in Fig 2.3.

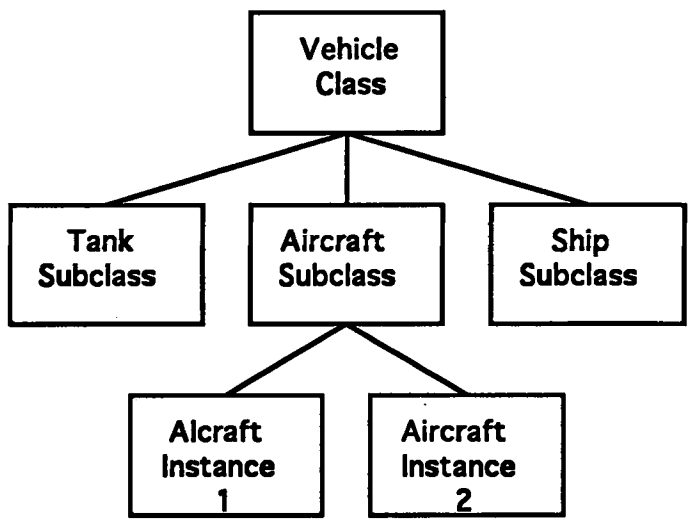

Fig 2.3 A vehicle frame hierarchy.

Since the class frames are generalisations they are placed higher in the hierarchy than the instance frames. The instance frames then inherit all or some of the properties stored in the class frames. Again, there is no standard meaning for the links within a frame-based system. The most common meanings are *is_a_kind_of* class link, which connect sub-class frames to their super-class frame, and *is_a* instance link, which connect the instance frames to their class frames. An example of a frame structure in the style of Ringland & Duce [126] is shown in Fig 2.4 to 2.6.

FRAMENAME	VEHICLE
SLOT1	SPEED: KNOTS OR MPH
SLOT2	ENGINE: PISTON OR JET
SLOT3	WEIGHT: KGS

Fig 2.4 Class frame VEHICLE.

FRAMENAME	AIRCRAFT
SLOT1	IS-A-KIND-OF: VEHICLE
SLOT2	ALTITUDE: FEET
SLOT3	TYPE: FIGHTER OR BOMBER

Fig 2.5 Subclass frame AIRCRAFT.

FRAMENAME	TORNADO
SLOT1	IS-A: AIRCRAFT
SLOT2	CREW: 2
SLOT3	AC-NUMBER: XL123

Fig 2.6 Instance frame TORNADO XL123.

2.5.2 Frame Inference Strategies

Since super-class slots are inherited by the sub-classes and the sub-class slots are inherited by the instance frames, inferences can be deduced about the lower level entries by following the inheritance links. Unfortunateiy, the links wiil have different meanings in different applications and each appiication may require its own set of inference strategies. However, Lucas and Van Der Gaag [102] describe two frame inheritance traversals, 'N' and 'Z' inheritance, which differ only in the order in which the frames are searched.

2.6 Knowledge Representation in Ada

The design of Ada started in 1974 and culminated in the publication of the language definition in 1983 [80]. The definition was the result of a United States Department of Defense initiative to replace the large number of programming languages, which were being used in the defence field, with a language which supported the principles of software engineering. Consequently, the language includes many facilities found in other languages plus additional features to support concurrent applications, modularity, information hiding and exception handling.

2.6.1 The Ada Programming Language

Ada is a very strongly typed, compiled, procedural programming language designed to cover a wide application domain. An Ada system is built from program units. Program units are either subprograms (procedures or functions), packages or tasks. Procedures and functions are sequential components similar to those found in other programming languages, whereas the package is the Ada modularisation component. The task is the language concurrent component and must be contained within one of the other program units. Subprograms and packages may be compiled separately to aid system development and maintenance. In addition, each program unit comprises a specification and a body which may be compiled separately.

Ada was designed to provide the ability to construct systems from independently produced software components in direct support of the software engineering modularity principle [80]. This requirement is satisfied by the package program unit construct. In addition, the principle of information hiding is supported by the Ada private type facility, which enables the detailed implementation of a package to be hidden from the user of a component. Furthermore, the requirement to reuse components in a number of diverse applications is supported by the generic construct. Finally, an exception mechanism is provided to enhance reliability and integrity.

2.6.1.1 The Generic Package

Thomas [141] shows how a software component may be formally specified as a generic abstract data type, and how a formal specification may be implemented using the Ada generic construct. An Ada generic specification is simply a parameterised package. The syntax of the generic part of a package is simple, and

comprises a list of formal generic parameters. The formal generic parameters may be object value names, type names or subprogram specifications.

The Ada generic specification provides a template from which instances can be created; this provides the ability to establish multiple and independent copies of the same component. Instance creation is a simple process called instantiation. Instantiation is achieved using the Ada construct *new* to associate actual generic parameters, provided by the user of the component, with the formal generic parameters defined in the generic package specification. Note that a package is a passive program unit which simply encapsulates other program units, although a package may have an active part, which is activated once at the point of elaboration during the run time process.

2.6.1.2 Ada Tasks

The Ada task is the means by which concurrent processing is achieved in an Ada system. A task specification contains the definitions of entry points through which other subprograms or tasks may communicate with the concurrent process. An entry point definition is very similar to the subprogram specification found in Ada, and procedure and function declarations in comparable programming languages such as Pascal and C. Each entry point is supported by an *accept* statement placed in the body of the task. If a program unit calls a task entry, and the task is not at the associated accept point, then the calling program unit waits. Conversely, if a task reaches an accept statement before a call is made to the associated entry, then the task waits. When the calling program unit has initiated an entry call, and the called task is at the associated accept statement, then the two program units synchronise by means of a rendezvous; during rendezvous data values may be passed between the task and calling program unit. On completion of the rendezvous, the two program

units disconnect and proceed concurrently. Note that this mechanism is excellent for implementing co-ordinating processes which need to suspend operation in order to consult other processes, and then reactivate at the point the suspension was initiated. The type of use envisaged here is that of a set of knowledge-based components co-operating to solve a complex problem. In this scenario, one can anticipate a control mechanism consulting a number of knowledge-based components in turn. Having elicited a response from one component, the component is suspended so that the response can be used as the basis of a query to another knowledge-based component to elicit agreement. Should this fail, then the previous component can be asked to reconsider its previous solution and find an alternative. This strategy can be used across multiple, and possibly diverse, knowledge-based components to establish a set of co-operating experts working toward some common goal.

2.6.2 Ada and Knowledge Representation

The Ada language was accepted as the ANSI MIL-STD-1815A-1983 and became the ISO standard 8652 in 1987. Furthermore, in some countries the language is mandated as the language to be used in military real-time embedded systems (USA), and specified as the preferred language for implementation in others (UK). In the early 1980s, knowledge-based techniques were not widely used in the context of real-time embedded systems. However, it was anticipated that there would be a need to do so; subsequently this proved to be a correct assumption. Consequently, an investigation was initiated in 1980 to assess the suitability of Ada for artificial intelligence applications [132]. The main problem in using a strongly typed procedural language such as Ada, which was designed to implement reliable, high integrity embedded solutions, to construct knowledge-based systems, is a clash in the way programs are developed. The implementation of real-time embedded systems require detailed analysis and design phases supported by strongly typed

languages for implementation; this involves extensive static and run time checks. Conversely, implementation of knowledge-based solutions normally involve weakly typed dynamic languages, which provide maximum flexibility for the experimental approach adopted by researchers. The report concluded that although Ada was found unsuitable as a general research programming language for artificial intelligence applications, a useful proportion of artificial intelligence programs can be re-implemented in Ada. It is interesting to note that the report identifies several extensions to Ada that would bring the language closer to what is required of an artificial intelligence language. These and other characteristics have now been addressed and are included in the first revision of the Ada language, Ada 9X [28] . For example, the extension of type definitions through inheritance, dynamic binding and pointers to procedures and functions. However, since Ada 9X compilers were not available during this research, Ada 83 has been used.

The advantages of using Ada in a knowledge-based systems context is that software engineering techniques can be used to build reliable and maintainable knowledge-based solutions. In addition, Naedai [111] shows that Ada can execute complex artificial intelligence algorithm-based programs 10 to 100 times faster than the equivalent LISP program. Naedal also concludes that an embedded knowledge-based solution contains only about 20 to 25% of code that can be classed as artificial intelligence algorithms, the rest is procedural, which further supports the use of Ada for the implementation of embedded knowledge-based components. He also points out that industry findings indicate that pre-defined Ada packages with specific artificial intelligence algorithmic capabilities appear to be a good way to perform rapid artificial intelligence prototyping, since it is straightforward to embed the components into a target system. This research supports these conclusions.

2.6.3 Current Ada Implementations

Since about 1985, when the first Ada compilers became available, there have been a number of experimental implementations of knowledge-based components in Ada. In particular Bobbie [14, 15], Baker [7] and Kilpelainen [88] have built logic components, Wallnau [144] and Wright [150] have built rule-based components, Scheidt [131] a semantic network and Waiinau [144] a frame component.

Wallnau's work [144] in particular is interesting since he introduces the idea of using abstract data types as the basis for building knowledge-based components. Wallnau concludes that "One tangible and significant advantage derived from using Ada was the relatively painless system integration phase"; this advantage was confirmed when it was found that the abstract knowledge type components were very easy to integrate with other conventional components. However, this research extends the ideas of using an abstract data type approach, introduced by Waiinau, in two ways: first, to include a control architecture in the set of knowledge-based components that can be used to integrate and co-ordinate component interaction; second, to provide the means of controlling multiple knowledge-based components that need to co-operate to solve a common problem.

2.7 Abstract Knowledge Types

The marriage between knowledge-based techniques and Ada offers the opportunity to develop reusable knowledge-based components in the same way that abstract data types are implemented. Since the knowledge-based components use the same concepts as abstract data types, the knowledge-based components are called abstract knowledge types(AKTs), where the data structures and operations of the abstract data type are replaced by the knowledge base and inference operations of the

abstract knowledge type. Fig 2.7 represents the Logic abstract knowledge type which is developed in Chapter 4. This shows a Knowledge Base, Inference Engine and Control mechanism encapsulated in an Ada package. The abstract knowledge type operations visible to a user component are Solve, Get_Result, Get_More and No_More.

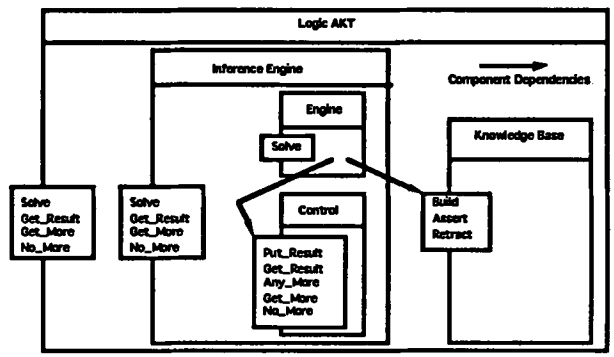

Fig 2.7 Logic abstract knowledge type architecture.

Any knowledge representation paradigm can be implemented in this way, and it is then a simple step to create a library of such components in exactly the same way as the current libraries of mathematical functions. System designers can then select the appropriate component(s) to model their domain knowledge. Since the components are generic Ada packages, there is no limitation to the number of independent abstract knowledge type instances that can be created; the instances can be of the same abstract knowledge type or a mixture of diverse abstract knowledge types. This type of approach is needed to support the designers of future large scale real-time systems, where they will be able to treat knowledge-based components in exactly the same way as conventional algorithmic components, leading to reliable, maintainable systems.

The use of abstract knowledge types gives the combined advantages derived from the abstract data types and knowledge-based techniques; these can be summarised as:

Abstract Data Type	Knowledge-based Techniques
Abstraction	Explicit knowledge representation
Modular solutions	Implicit inferencing algorithms
Information hiding	Easily modifiable knowledge
Rapid development through reuse	Incremental growth of knowledge
Application independent	Diverse representations
Implementation independent	Expert consultation
Multiple instantiation	Prototyping in application domain
Formal specification	
Enhanced reliability and integrity	

Fig 2.8 Advantages of using abstract knowledge types.

2.8 Summary

This chapter describes the knowledge representation paradigms used to test the thesis set out in Chapter 1; these are logic, rules and frames. In order to implement these components in real-time embedded systems, it is highly likely that this will have to be done in the Ada programming language, since some countries have mandated Ada as the implementation language for real-time embedded systems (USA) and others specify the language as preferred (UK). However, the marriage of Ada and knowledge-based techniques provides the opportunity to implement these components in a similar way to abstract data types; the knowledge-based components have been called abstract knowledge types. Consequently, this approach results in components exhibiting advantages taken from the application of abstract data types with those from using knowledge-based techniques. Finally, it is proposed that a library of abstract knowledge types should be developed in order to give system designers the opportunity to match complex problem domains to the most appropriate knowledge representation paradigm, without the need for extensive development of the knowledge-based components. This presents the opportunity of using a prototyping approach in the real-time application environment.

Chapter 3

Integration of Knowledge Representation Paradigms

3.1 Introduction

Chapter 1 identified future applications which will need to use diverse and possibly multiple knowledge representation paradigms in order to produce the most appropriate model of the complex problem domains being encountered in the development of real-time embedded systems. Chapter 2 described a selection of knowledge representation paradigms and proposed that all such paradigms be encapsulated as abstract knowledge types in a similar way to abstract data types. In addition, Chapter 2 recognised that some countries have mandated the Ada programming language for the implementation of real-time embedded systems, while others have promulgated a preference for Ada in this context. The aim of this chapter is to review the current methods of integrating knowledge representation paradigms and to propose an approach to integration in real-time embedded systems using Ada.

3.2 Language Integration

Many attempts have been made to integrate programming language paradigms. For example POPLOG, which combines three languages in a single environment; POP11, Prolog and LISP. More recently, languages have been extended to include the

object-oriented paradigm, for example Prolog ++, CLOS, C++ and Ada 9X. Although this approach adds the concepts of class hierarchies, inheritance and dynamic binding, the languages still have limited knowledge representation primitives and much low level work is necessary to integrate more paradigms. Consequently, integration at this level is not recommended. However, Ada 9X will be used in future work to build a library of operational abstract knowledge types.

3.3 Hybrid Tools

In a knowledge representation review paper, Brachman [19, 20] notes that by 1980 only a few people had experimented with approaches to integrate different knowledge representation paradigms. However, by the mid-1980s much work had been done to integrate multiple knowledge representation paradigms in what have become known as hybrid development environments.

Daniel and Haugh [44] carried out an extensive study of hybrid tools in a search for a suitable implementation vehicle for the data fusion application implemented by Miles [106]. The resulting report identifies common characteristics for these systems when considered for use in developing the data fusion system. The common advantage is the availability of powerful graphical interfaces which support the prototyping approach favoured by artificial intelligence researchers. However, the main disadvantages are:

- Lack of real-time support
- Lack of extensive monitoring and debugging facilities
- Lack of explicit control knowledge
- Inflexibility - no possibility to extend the language apart from LISP macros
- Slow execution speeds

In an experiment to test the use of hybrid systems in data fusion, Miles [107] showed that the hybrid system was 200 times slower than a bespoke Ada solution. Furthermore, the most common implementation language for the hybrid tools is LISP, which fails the Ada mandate. In addition, although the tools do integrate different knowledge representations, these are limited to two or three, usually frames and rules or objects and rules. Consequently, apart from the attractive sophisticated prototyping environment, pure hybrid tools are not ideal for the implementation of diverse knowledge representation paradigms intended for real-time embedded applications.

3.4 A Hybrid Tool with Bespoke Code

Gillies [62] recognises the advantage of using a hybrid tool to prototype an application and proposes that the resulting prototype be re-engineered in Ada to achieve the desired requirement. To this end, Gillies presents a case study of a military project that developed a decision support system for the organisation for combat, which was constrained by the Ada mandate. The company used a hybrid tool to establish the feasibility of the solution and then transformed the knowledge-based prototype, first into C and then into Ada! No reason was given as to why this double transformation was needed. This approach is not recommended because of the time overhead and the possibility of introducing errors during each transformation. However, an abstract knowledge type library would have reduced the risk.

3.5 A Hybrid Tool with Code Generation

Hintz [76] notes that "While it is possible to create a diagnostic expert system directly in Ada, it would be extremely time consuming to do so. The knowledge

representation, inference engine and man-machine interface support software is just not available in Ada and would have to be created". It is this problem which the abstract knowledge type approach addresses. However, since Hintz did not have such a library of components he turned to a hybrid tool with a automatic code generator to solve his problem.

ART-Ada™ is an extension of the ART™(Automated Reasoning Tool) hybrid environment which provides rule and frame representations plus procedural representation via Ada. The knowledge-based aspects of an application solution are first developed in the ART-Ada language - a Common LISP like syntax - which is then transformed by a code generator into Ada. The generated Ada code is then compiled and linked in the context of an ART-Ada kernel, together with other application Ada program units. The main advantage cited for using this approach is that since the knowledge representation paradigms are not available in Ada, then it is less time consuming to use this cross compilation technique. Hintz concludes that this work is the first step towards a "pure" Ada expert system; the abstract knowledge type approach could form the basis of the "pure" approach. Finally, although the code generator is an improvement on the bespoke solution, a user is restricted to frames and rules.

However, Collard [31] raises some software engineering issues regarding this approach; these are:

- Performance. Whether the application code translated into Ada should perform as fast or as slow as its implementation in LISP.

- Real-time Accommodation. For translation of real-time codes it appears highly unlikely that such a translator could insure the same real-time characteristics found in the LISP version in the Ada version.

- Evolution. If new features are incorporated in the Common LISP standard, then the translator must be updated and re-verified. This task may be equal in cost to the one of building the original translator.

- Maintenance. The Ada code compiled by the translator is most likely indecipherable by software engineers unless it is built to provide comments on the translation process.

Consequently, an alternative 'pure' Ada approach is preferred.

3.6 Blackboard Architectures

Since the Ada language was designed to be extended by the addition of packages, an obvious way to solve the integration issue is to simply add an integration component(s) to the abstract knowledge type library. The blackboard architecture can form the basis of such a component.

3.6.1 Blackboard Origins

Numerous analogies have been used in the literature to describe the blackboard concept. The most common is that of a group of experts gathered around a conventional wall mounted classroom blackboard; the group is trying to solve a problem and uses the blackboard to record ideas and partial solutions. As the solution evolves, information placed on the blackboard by one expert triggers other experts to respond with new information; this may involve addition, deletion or change to the partial solution already recorded on the blackboard. Consequently, the opportunistic recording of contributions on the blackboard, by each expert, leads to an incremental evolution of the problem solution.

The original blackboard system is described by Erman and Lesser [57]. Although the blackboard concept was initially applied to the problem of speech understanding, the Hearsay-II system, the architecture was developed to provide a general system-building framework for co-ordinating independent processes. Consequently, other researchers were able to apply the concept to many different applications. In particular, early investigations included multi-sensor interpretation, protein-crystallographic analysis and image understanding. Since then, the concept has been applied to a multitude of problems. The literature search identified over 100 explicitly named blackboards with numerous other applications which used blackboards in their implementation.

The Hearsay-II system recognises connected speech using knowledge-based techniques. The analysis of speech signals is undertaken by a series of knowledge sources, processes which represent diverse acoustic and linguistic knowledge. At each stage of the speech analysis, potential partial solutions are encapsulated as hypotheses and entered on a global hierarchical data structure, called the blackboard. The blackboard has several levels, each level representing a particular intermediate stage in the speech understanding process. Each level is seen as an abstraction of the next lower level. Taken as a whole, the levels can be thought of as a plan to solve the speech understanding problem.

Each acoustic and linguistic knowledge source was designed to transform data between two blackboard levels or within a single blackboard level. Knowledge sources are independent condition-action modules which are only allowed to communicate via the blackboard. When a knowledge source condition is satisfied, by the arrival of an hypothesis on the blackboard level to which the knowledge source has been assigned, the knowledge source action is activated; this action generates an hypothesis, or modifies an existing one, on a different or the same blackboard level.

The change in the blackboard state results in other knowledge source conditions being satisfied, and other knowledge source actions subsequently activated. In this way, the speech solution is incrementally established on the blackboard, by the knowledge sources, in an opportunistic manner.

A control mechanism is provided in order to schedule the knowledge sources. On each processing cycle there may be several knowledge sources whose conditions have been satisfied by the current state of the blackboard. Consequently, a scheduler has to establish a priority for each knowledge source, and select the one with the highest priority value to apply to the blackboard.

The Hearsay-II project established the blackboard system as a powerful, generally applicable problem solving architecture.

3.6.2 The Blackboard Model

The generally applicable blackboard architecture, abstracted from Hearsay-II, is described by Nii [115] as the blackboard model. The model describes three key blackboard components: first, a global hierarchical data structure, called the blackboard, on which linked partial solutions are recorded; second, knowledge sources, which encapsulate diverse problem-specific knowledge; third, an opportunistic control strategy used to establish which knowledge source is the most appropriate to apply to the blackboard. The knowledge sources respond opportunistically to changes in the blackboard state and either create, delete or amend the blackboard partial solutions. In this way the solution space, depicted on the blackboard, incrementally progresses towards a solution of the problem.

Similarly, Craig [43] defines a blackboard architecture to have four key elements;

first, entries, which are intermediate results generated during problem solving; second, knowledge sources, which are independent, event driven processes that produce entries; third, the blackboard, a structured global database which mediates knowledge source interactions and organises entries; four, an intelligent control mechanism which decides if, and when, particular knowledge sources should generate entries and record them on the blackboard.

In general, the two definitions appear very similar. For example, Nii describes a global hierarchical database containing linked partial solutions, whereas Craig defines entries on a structured global database. This suggests that Craig anticipates the possibility that a blackboard may not necessarily be hierarchical. In addition, both definitions included a control element, however, Nii is very specific in requiring an opportunistic control strategy. Craig on the other hand requires the control element to be intelligent and the knowledge sources to be event driven, but does not consider opportunism as an essential requirement. Indeed, Craig goes to great lengths to argue that opportunism is "a species of control strategy and not a consequence of the blackboard architecture" and as such opportunism should be considered as just one of a number of control strategies that may be applied by a developer. In both definitions the nature of the knowledge sources is not prescribed. In general, this is true throughout the literature.

3.6.3 The Blackboard Framework

Although the blackboard model is ideal for gaining an initial understanding of the blackboard concept, it does not provide the detail from which a practical system can be developed. Consequently, Nii [115] expands the model to provide a more detailed blackboard framework; this is shown in Fig 3.1.

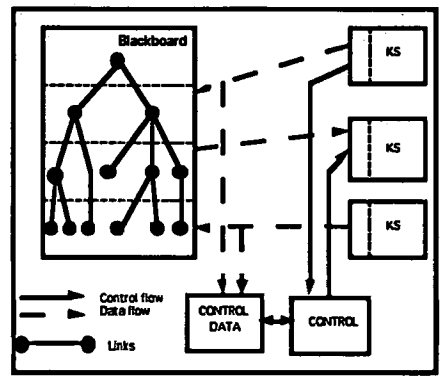

Fig 3.1 The blackboard framework.

The abstracted framework reveals a number of characteristics which may be used to define a blackboard system. In addition, Craig [43] gives an eleven point summary of a blackboard architecture. The two views are presented for comparison in Fig 3.2.

The definitions appear very similar, but closer analysis reveals some fundamental differences. It is the generality of the statements given by Craig which makes his definition different from that of Nii. In particular, Nii presents a mental image of the blackboard which is simply hierarchical, the case in Hearsay-II. However, Craig believes that the relationship between entries on the blackboard may be more complex than this suggests. Furthermore, Craig sees the knowledge sources as defining the relations between blackboard levels. Consequently, Craig uses relational attributes, abstracts/refines and adjacent-to, to define the vertical and horizontal structure of the blackboard.

Both definitions present a knowledge source as having a condition and action part. In addition, Nii defines a knowledge source as being represented by procedures, rules or logical assertions. However, Craig does not make any assumption about the nature of the action part of the knowledge source, other than it performs computation and generates blackboard modifications.

Nii [115]	Craig [43]
The blackboard holds computational and solution-state data. The blackboard consists of objects from the solution space. The objects and their properties form a vocabulary of the solution space. The relationships between objects are denoted by named links. The objects on the blackboard are hierarchically organised into levels of analysis. The blackboard can have multiple panels.	Problem solving activity generates a set of intermediate results which are represented as objects with attributes and values. The objects are called entries. All entries are recorded in a global database called a blackboard. Entries may have user-specified relationships with other entries. All entries have the relation attributes: abstract/refines and adjacent-to. These attributes define the vertical and horizontal structure of the blackboard. The blackboard may have additional, user-specified, structure.
Knowledge sources contribute information that will lead to the solution of the problem. Each knowledge source is responsible for knowing the conditions under which it can contribute to a solution. Knowledge sources modify only the blackboard or control data structures. Knowledge sources are represented as procedures, rules or logical assertions. Knowledge sources respond opportunistically.	Independent knowledge-representing processes, called knowledge sources, generate, modify and record entries on the blackboard. Each knowledge source has a condition and an action. The condition matches hypothetical configuration of entries on the blackboard, performs computation and is a predicate. The action performs computation and generates blackboard modifications.
There is a set of control modules that monitor the changes on the blackboard. Various kinds of information are made globally available to the control modules. The focus of attention indicates the next thing to be processed. The solution is built one step at a time. The problem solving activity is iterative. Criteria are provided to determine when to terminate the process.	An intelligent scheduler determines which triggered knowledge source(s) should execute its(their) action(s). The scheduler can base its decisions on user-determined criteria such as the characteristics of the triggered knowledge source, the utility of the proposed action, information about the general blackboard state, characteristics of the problem, or information about previous control decisions.

Fig 3.2 A comparison of blackboard characteristics.

3.6.4 Blackboard Structures

Starting with the initial development of the blackboard system for Hearsay-II, numerous refinements have been proposed to the architecture to meet the needs of particular applications, and to provide generally applicable blackboard shells.

3.6.4.1 Single Monolithic Blackboards

Initially, the blackboards were monolithic linked structures. One of the first detailed accounts of the blackboard was given by Erman & Lesser [57] where they describe the blackboard as a "..uniform and integrated multi-level structure". The levels comprise hypothesised elements representing the dynamic state of a problem solution and are shown in Fig 3.3.

Blackboard Level Names	Hypothesis Representation
Conceptual	Concepts
Phrasal	Syntactic Elements
Lexical	Words
Syllabic	Syllables
Surface-Phonic	Phoneme Like Units
Phonetic	Phonetic Description
Segmental	Acoustic Segments
Parametric	Acoustic Signal Data

Fig 3.3 The Hearsay-II blackboard levels.

Hypotheses at one level are related to hypotheses on other levels, usually adjacent, by links; a lower level hypothesis is said to support an abstracted hypothesis at a higher level. The Hearsay-II blackboard is split into levels in order to:

- Mirror the decomposition of the knowledge into knowledge sources.
- Limit the scope of the blackboard available to each knowledge source.
- Permit efficient sequencing of the knowledge source activities.

- Provide a hierarchy of abstraction with each level holding a different representation of the problem.
- Permit new levels to be added as new sources of knowledge are designed.

This structure is used by Miles [106] to achieve data fusion in a Naval scenario. Three blackboard levels are used in this application: first, sensor data is entered on the lowest level of the blackboard; the middle level represents multi-track hypotheses derived from like sensor hypotheses located on the lowest level; the top level represents vehicle hypotheses derived from multi-sensor, multi-track hypothesis correlations on the middle level. This strategy is illustrated in Fig 3.4.

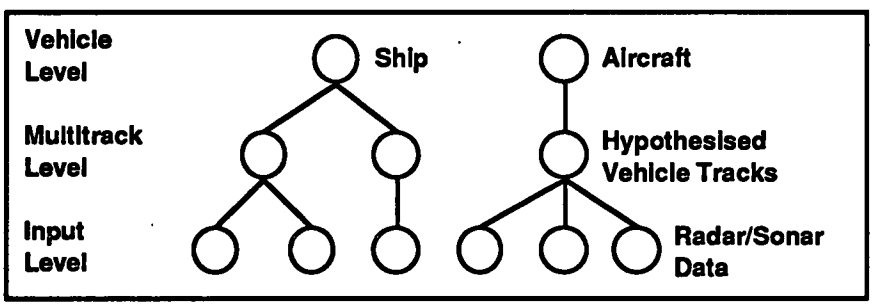

Fig 3.4 The Naval data fusion blackboard.

The advantage of the single monolithic blackboard approach is simplicity in understanding and implementation, coupled with generality of application.

3.6.4.2 Partitioned Single Blackboards

As single monolithic blackboards become more complex, there is a natural tendency to partition the blackboard into functional areas.

Hayes-Roth [69] describes an architecture, to support research into 'Planning', that partitions a single blackboard into the five 'planes' shown in Fig 3.5. Each

plane contains several linked hierarchical levels; this was found useful since the blackboard partitions are used to model Hayes-Roth's assumption that people make decisions at different levels of abstraction. In addition, links are established between planes.

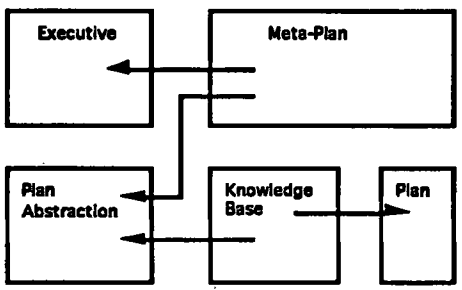

Fig 3.5 Blackboard planes.

Craig [40] describes an architecture where the blackboard is subdivided into an arbitrary number of partitions shown in Fig 3.6.

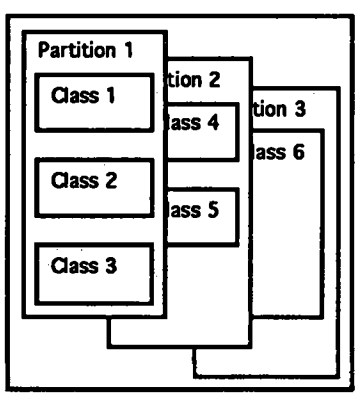

Fig 3.6 Blackboard partitions and classes.

Each partition is then divided into an arbitrary number of classes. Craig claims that this architecture encourages the application to separate control and problem solving activities which leads to more understandable solutions. In addition, this architecture has two other interesting properties: first, no restriction is placed on the relationships between partitions or between classes; second, the partitions and classes can be created dynamically. These characteristics make this strategy

application independent, an important step in the evolution of blackboard architectures, since early blackboards have been designed with a particular application in-mind.

The idea of nested blackboards is described by Hayslip & Rosenking [74] for real-time planning in high speed combat aircraft and is illustrated in Fig 3.7.

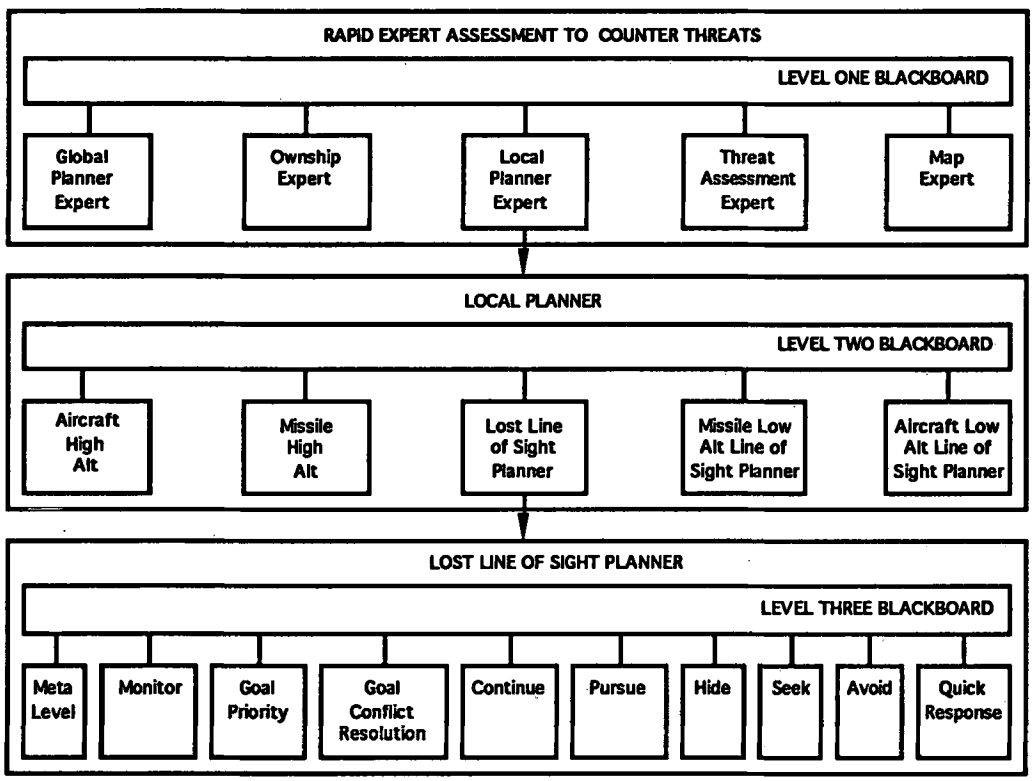

Fig 3.7 Nested blackboards.

The nested blackboard is designed for operation in applications requiring fast, concurrent processing. In this architecture, a nested blackboard acts as a cache to local knowledge sources which represent an aircraft's new state, while the outer blackboard depicts the previous state of the system; only complete sets of data are passed from the lower blackboards to the higher level blackboard. The advantage of this is that, on interrupt, the old aircraft state is easy to find on the outer blackboard. The paper concluded that this architecture is very versatile.

These few examples of partitioned single blackboards show that the strategy is useful in enhancing the model of particular applications and allows blackboard implementations to be built which are flexible and application independent.

3.6.5 Multiple Blackboards

With a view to increased modularity and the possibility of building a blackboard architecture across distributed systems, researchers have developed multiple independent blackboards. A number of notable examples are describe to illustrate the strategy.

Erman et al. [58] uses a relational database to represent separate control and domain blackboards in Hearsay-III, a domain independent architecture designed to explore problem solving in user applications. The separation of control knowledge from the domain knowledge proved to be an important step in the development of blackboard architectures, in that explicit reasoning can be applied to blackboard control as well as to the domain knowledge. The work on Hearsay-III concluded that the use of separate blackboards for the problem domain and control reasoning gives a flexible approach toward developing a set of diverse scheduling algorithms, so simplifying this complex aspect of problem solving; the application can select an appropriate scheduler from a pre-defined set.

The generic blackboard development system GBB, described by Corkill [35], was built in order to reduce the time required to implement specific applications, and to improve the efficiency of the resulting implementation. The GBB blackboard structure exhibits the hierarchical characteristic of previous blackboards, but the structure is formed from blackboard spaces or multiple blackboards comprising blackboard spaces. In addition, because of the requirement to provide a generic

architecture, the blackboard spaces can have different dimensionality, unlike previous architectures where the dimensionality of each level is one[1]. In addition, the architecture separates the definition of the blackboard objects from that of the database manipulation operations as illustrated in Fig 3.8. Consequently, management of the database can take place without affecting the blackboard objects.

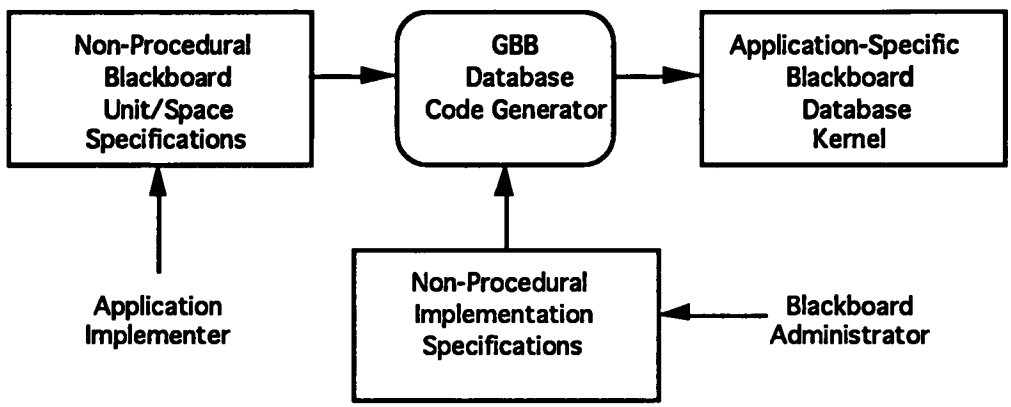

Fig 3.8 The GBB database subsystem.

A recursive agent blackboard model was used by Yoshida & Hino [151] as an object framework for pattern recognition. In this model blackboard/agent (knowledge source) pairs are represented by objects. Each blackboard/agent object can act as a blackboard for its inner agents and/or act as an agent to its outer blackboard. The resulting structure forms an object inheritance hierarchy modelling a divide-and-conquer strategy. Each agent in the hierarchy searches only its local blackboard

---

1 Dimensionality is a measure of the number of different areas associated with each blackboard space. A dimension of one indicates that the space is a single area into which all objects are placed. A dimension of three indicates that there are three different areas in the space into which objects may be placed. An application may define the number of blackboard spaces and the dimensionality of each blackboard space. The space dimension may be ordered, for example a time dimension, or enumerated, for example vehicle types, or both ordered and enumerated.

region, but co-operates with other agents in order to search the entire hierarchy. The advantage of this approach is to improve the efficiency of the pattern recognition process.

These examples show that the strategy of implementing multiple blackboards further enhances the construction of application independent implementations, can be used to improve efficiency, but, more importantly, the strategy forms the basis for providing a means of explicitly reasoning about control.

3.6.5.1 Distributed Blackboards

A natural extension to the idea of multiple blackboards is to spread the blackboards across a distributed system.

Lesser & Corkill [99] use a modified Hearsay-II blackboard architecture to explore the issues associated with building distributed problem solving networks. A remote sensor vehicle tracking application was chosen as the basis for experimentation. A simulation of the vehicle tracking network was set-up where each network node, shown in Fig 3.9, is represented by a complete Hearsay-II blackboard system; the architecture has been modified to permit inter-node communication, goal-directed planning and meta-level control.

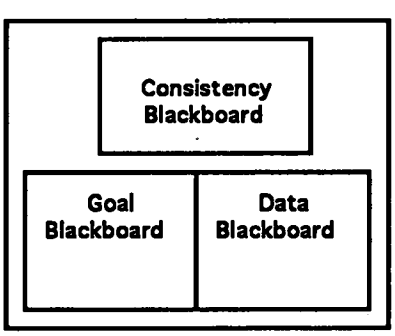

Fig 3.9 A distributed node.

The advantage of this approach is that each node has the full processing capability of a blackboard system configured to solve a local problem.

An extended blackboard architecture, designed to operate in a multiprocessor environment, is described by Ensor & Gabb [56] and shown in Fig 3.10.

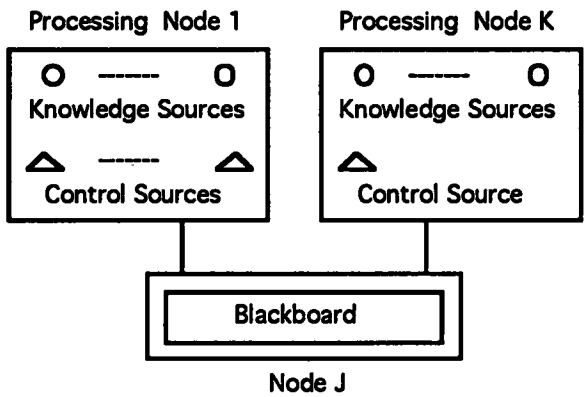

Fig 3.10 Distributed knowledge sources.

Knowledge and control sources, are distributed over different processors and access a central blackboard using transactions; a transaction manager is associated with the blackboard. Since transactions are asynchronous, the transaction manager uses read and write locks to maintain blackboard consistency. The model also permits direct knowledge source communication, a major deviation from the original blackboard model.

A blackboard architecture comprising encapsulated level managers was used by Saxena [130] to investigate distributed blackboard knowledge representation issues; each level manager is an independent process that executes synchronously. A level manager contains the components shown in Fig 3.11. These can be distributed in any way and are not limited to the linear hierarchies found in earlier systems.

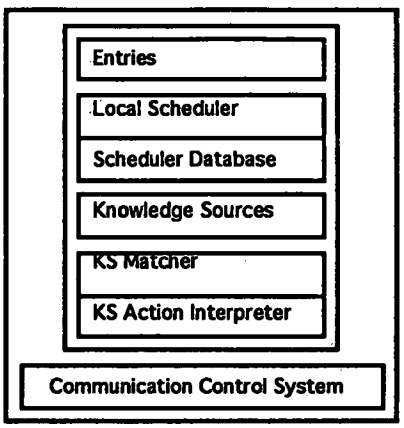

Fig 3.11 Level manager.

3.6.6 Blackboard Control Strategies

Knowledge source selection is usually event driven, Hayes_Roth [70], that is, changes to the blackboard result in the selection of the next knowledge source to apply to the blackboard. Once the knowledge source has been selected, the generation of entries on the blackboard may proceed from a lower to a higher blackboard level, or from a higher to a lower blackboard level; Erman and Lesser [57] identifies these processes as synthesis and analysis respectively. Furthermore, the knowledge sources themselves may be goal or data directed. In a goal directed environment knowledge sources are chosen to satisfy a goal. However, in a data-directed environment knowledge sources are chosen in order to process a blackboard entry. During each problem solving cycle the control mechanism may have to choose between several knowledge sources that are ready to make a contribution to the solution. Consequently, various scheduling schemes have evolved.

3.6.6.1 Implicit Control Strategies

In early blackboard architectures the control knowledge is embedded in the

application code. Consequently, these implicit control strategies are difficult to understand and are not generally applicable.

3.6.6.2 Knowledge Source Condition/Action Lists

In the original blackboard system, Hearsay-II [59], control is achieved through a blackboard monitor and scheduler. The blackboard monitor tracks the changes made to the blackboard, and maintains a scheduling queue containing references to the knowledge source conditions and actions which are able to make a contribution to the solution. On each processing cycle, the scheduler calculates the priority of each of the queued activities and executes the activity with the highest priority. The condition activities, if true on evaluation, result in knowledge source actions being added to the scheduling queue. However, executing a knowledge source action results in changes to the blackboard.

3.6.6.2.1 Blackboard Entry Event Lists

The HASP system [116] establishes control by maintaining five event lists which record pending blackboard activities; a rule based-event manager is allocated to each of the event lists. On each scheduling cycle, a rule-based strategy knowledge source decides which event manager is the most appropriate to apply to the problem partial solution. The activated event manager then selects the most appropriate event, from its associated event list, and a knowledge source(s) to process the event.

3.6.6.3 Explicit Control Strategies

In later schemes, the control knowledge is made explicit in order that applications

can reason about control strategy in a similar manner to the way the application reasons about its domain knowledge. The advantage of this approach is that control strategies can be modified, or a different strategy selected, by an application as a solution evolves.

3.6.6.3.1 Hierarchical Control

Nii et al. [112, 114], describe two different applications where a three level hierarchical control strategy is used. In this strategy, decisions are achieved by descending a hierarchy of control knowledge sources. Strategy decisions are taken by a top level control knowledge source to decide which region of data should be analysed next; having selected the region of interest, the middle control knowledge sources select the domain knowledge source, which resides on the lower level of the control hierarchy, that is to access the hypothesis hierarchy. Two separate blackboard planes are used; one blackboard plane contains the control hierarchy and the second blackboard plane, the hypothesis hierarchy.

Engelmore & Terry [54] and Terry [140] describe a blackboard control strategy where the control decisions are derived from a hierarchical production system (HPS). Each level in the HPS contains an explicit set of control rules. A control cycle starts by consulting the control rule in the top level of the HPS. The control rules at each level are then used to select a control action at the next lower level. The control process continues until the lowest level of the HPS is reached. The lowest level of the HPS contains the rule sets that, when selected, contribute to the solution by changing the blackboard.

3.6.6.3.2 A Blackboard Control Model

Hayes-Roth [70] recognises the problem of control as being fundamental to intelligent systems. Consequently, in the blackboard control model proposed by Hayes-Roth, the control knowledge is made explicit in the form of control knowledge sources and a control blackboard; the control blackboard entries record solutions to the control problem on blackboard levels which are domain Independent. Consequently, the model permits reasoning about both the domain and control problems, unlike earlier systems where reasoning is limited to the domain knowledge, because the control mechanism is implicit. Moreover, different control strategies, for example data-driven or goal driven inferencing, can be chosen dynamically, as required, by the control system; this is in contrast to the fixed implicit system of Hearsay II. The model also permits a choice between activating domain or control knowledge sources. Furthermore, by recording control decisions as entries on the control blackboard, past control decisions can be used to influence the choice of subsequent control actions.

This work was an important advance in solving the blackboard control problem. It lays down a domain independent theory for blackboard control.

3.6.6.3.3 Decentralised Control Across the Blackboard

Craig [43] developed a general purpose architecture in which the control scheme is distributed amongst the blackboard levels. In this architecture a blackboard level comprises a domain database and associated domain knowledge sources encapsulated in a level manager. The level manager relies on a local controller to schedule the encapsulated knowledge sources. In addition, the level managers communicate using the concept of ports, defined in a levels' interface, and channels which link the

ports. Craig claims several advantages for this architecture:

- The encapsulated levels enhance blackboard modularity.
- Enhanced modularity provides a better base for applying concurrency and building distributed blackboards.
- The communication model permits a more flexible structure to be created than is possible with the conventional hierarchical blackboard.
- Local controllers can use different control strategies.
- The architecture can be easily extended by adding new level managers.

3.6.6.3.4 Distributed Control with a Central Blackboard

Elfes [53] describes a distributed control system for use in autonomous mobile robots. In this architecture control is distributed over a processor network as expert modules; modules communicate via messages through a central blackboard. Each module has a Master and a Slave process: the Master process schedules the Slave process and acts as the interface between the Slave process and the blackboard; the Slave process provides the domain computations needed to update the blackboard.

3.6.6.3.5 Distributed Control with Distributed Blackboards

As described earlier, Saxena [130] distributes the blackboard and knowledge sources across multiple processors. In addition, control is also distributed, resulting in a control system that has both local and global control strategies. Consequently, each level manager is able to carry out local control based on its local

state, whereas global control is achieved by message-passing; this results in a completely distributed control system.

3.6.7 Blackboard Implementation Languages

LISP appears to be the most popular language for the implementation of blackboard architectures, since most blackboards have been contrived in a research environment. Very little work seems to have been carried out to look at implementation issues when faced with the problem of building blackboard architectures which are to be embedded in real-time environments.

A notable exception to this is the work done by Miles [106]. In his research, Miles compares the implementation of a blackboard architecture implemented in Ada with that of the same problem implemented using a production system and a hybrid tool, comprising both production and procedural components. Using thirty minutes of operational test data, Miles recorded an improvement in processing time using Ada of 200 times when compared with the production system, and 50 times when compared with the hybrid solution. It is interesting to note that Miles did not have access to a library of knowledge-based components, but was limited to the use of rules as the only form of representation in the Ada solution. Furthermore, the rules are embedded directly into the Ada code rather than being separate, as would normally be expected in a knowledge-based solution. This obviously contributes to the decrease in processing time of the Ada implementation, since a rule matching phase is not used, but in doing so the advantages of having a separate knowledge-base are lost. It is this work that stimulated the idea of providing a library of knowledge-based components that will make the design and implementation of real-time knowledge-based systems easier to accomplish, and of using a blackboard as an integrating architecture.

3.6.8 Blackboard Applications

Engelmore and Morgan [55] suggest four reasons for using a blackboard architecture in an application, rather than other architectures such as the rule based paradigm. These are:

- Modularity — The architecture is inherently modular.
- Dynamic Control — A wide range of control strategies are available.
- Efficiency — Control policies can focus effort.
- Concurrency — The modular nature of the knowledge suggest possible concurrent operation.

To these Saxena [129] adds:

- Flexibility — The blackboard structure has been adapted to fit many applications.
- Extensibility — The number of levels and the number of knowledge sources can be extended as a design evolves.
- Generality — Blackboard systems have been applied to a wide variety of applications.

Having studied many example blackboard applications, Engelmore and Morgan [55] also suggest that blackboard applications appear to have one or more of the following attributes:

- Many specialised and distinct kinds of knowledge.
- Integration of disparate information.
- A natural domain hierarchy.
- Continuous data problems.
- Applications with sparse knowledge/data.

Furthermore, Laasri et al. [93] suggest that it is preferable to use a blackboard architecture in applications where:

- It is necessary to analyse a very large amount of information.
- The analysis of the field of applications can be divided up into different abstraction levels.
- The problem requires the collaboration of several experts.
- Opportunist strategies must be utilised.

Note that the literature search identified over 100 explicitly named blackboards, with many others using blackboards as part of their implementation.

3.6.9 Knowledge Sources

There is very little discussion in the literature regarding the nature of knowledge sources, apart from occasional reference to the domain knowledge being procedural or heuristic. In addition, little appears to have been written about how to construct the knowledge sources from multiple and diverse knowledge representation paradigms.

In the papers that do make reference to knowledge source implementation, it appears that the choice of representation in blackboard systems is limited, and is usually based on the language and/or the environment in which the blackboard is implemented. However, a few authors explicitly identify the need for diverse knowledge representation in knowledge sources. In particular, Baum [10] identifies knowledge source representation as being a key research issue and concluded that "blackboard shells are more versatile when they can support a variety of knowledge source representation schemes". He noted that "This is advantageous, since having a good mapping from problem domain to the

implementation increases the clarity of the application". In addition, Baum [11] also identifies one of a number of considerations in the design of blackboard systems as being the need for representational adequacy. That is, the blackboard should allow adequate encoding of problem domain information and, since there is probably no single representation paradigm which is appropriate for all problems, the blackboard should support a set of different representation paradigms. This suggests a need for a set of diverse application independent knowledge-based components that can be used when constructing the domain knowledge sources.

3.6.10 Choice of Blackboard Architecture

This investigation shows that the blackboard is an adaptable, generally applicable problem solving architecture which is suitable for use in integrating instances of diverse abstract knowledge types in a wide variety of applications. In addition, its inherent modular structure make it ideal for implementing as an abstract component to be placed in the library of abstract knowledge types. Consequently, the blackboard framework described by Nii and used by Miles was chosen as the integration model for this research. The were two main reasons for this choice: first, the model is easy to understand, so potentially easy to implement as a generic component; second, the model is familiar, since a two year project involving the implementation of the data fusion process postulated by Miles, had been completed in five different programming languages. This work compares data fusion implementations in Ada, C and Smalltalk 80 with implementations in two new languages, Rekursiv C and Lingo, both designed for a new novel architecture called Rekursiv [66].

3.7 Summary

The original blackboard architecture comprises a single hierarchical data structure, implied control algorithms and domain knowledge encapsulated in knowledge sources. Subsequently, researchers extended this basic model to include:

- Single blackboards split into separate panels.
- Multiple independent blackboards.
- Explicit control knowledge enabling reasoning about control strategies.
- Use of multiple blackboards to represent domain and control knowledge.
- Distributed knowledge sources accessing a central blackboard.
- Distributed control across blackboard levels.
- Distributed blackboard systems across multiple processors.

This chapter has shown that the blackboard, through diversity of application, is considered a generally applicable problem solving architecture. Furthermore, although there have been many variations on the original blackboard design, there is general agreement that the architecture has three main components: a blackboard, on which the solution evolves; knowledge sources, which encapsulate domain knowledge and make the evolutionary changes to the blackboard; a control mechanism, which decides which knowledge source should have access to the blackboard at any given time. Although much research has been carried out to explore the potential of the blackboard and its control strategies, little work has been recorded regarding the implementation of the knowledge sources. The purpose of a knowledge source is to encapsulate domain knowledge. Since the domain knowledge in complex real-time systems will require the use of diverse and possibly multiple knowledge representation paradigms, integration issues need to be addressed. Consequently, the blackboard model described by Nii and used by Miles, with event list scheduling, is used as the integration component in this research.

Chapter 4

Implementation of a Logic Abstract Knowledge Type

4.1 Introduction

Chapter 2 identified logic, which is often used in the literature to introduce the concept of knowledge representation, as being one of the most common representation paradigms. Furthermore, the paradigm is widely used through the use of PROLOG language implementations in which many applications are currently being implemented, Roth [128]. Although the full power of such a language is not appropriate for an embedded real-time component, the backward chaining, pattern matching and backtracking characteristics of such an implementation offers attractive advantages for use in complex knowledge-rich real-time environments. Consequently, the aim of this chapter is to describe how a logic abstract knowledge type, providing restricted PROLOG functionality and a control mechanism to permit component co-operation, was implemented.

4.2 Knowledge Base Data Structures

Some of the abstract principles of PROLOG implementation are discussed in Kluzniak & Szpakowicz [90], where the main issue revolves around the choice for implementing instance variables. Two methods are described: first the non-structure sharing method in which a copy of a clause structure and its variables is

created to represent each matched instance; second, the structure sharing method where only the clause variables are copied, with each instance sharing a single representation of the clause structure. Since structure sharing is inherently more efficient than non-structure sharing, structure sharing was the method chosen for use in this experiment.

4.2.1 Knowledge Base Architecture

This implementation provides facilities to represent PROLOG facts, rules and structures. Since the syntax of PROLOG is relatively simple, the data structures were chosen so that the internal representation is as close to the external representation as possible. The three nodes shown in Fig 4.1 are used to build the internal representation of a PROLOG knowledge base.

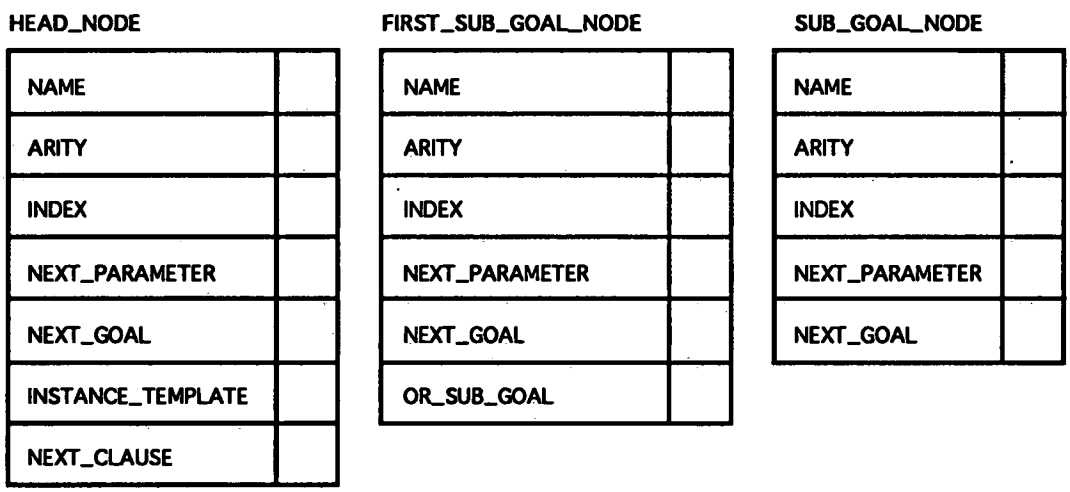

Fig 4.1 The logic knowledge base nodes.

All nodes share common fields for Name[1], Arity, Index, Next_Parameter and Next_Goal. The Name of each atom or variable is stored as a dynamic string [16],

[1] Naming conventions are those used in the code Annexes.

followed by the Arity[2] of the node entry. The Index field records the position of a variable in the Instance_Template Instance_Variables_Record; a zero value indicates that an atom is being represented, which does not require entry in the Instance_Template Instance_Variables_Record. For example, Fig 4.2 shows the head node of the rule p2 in the clause

p2(W, X) :- p3(W, Y); p4(c, X, Y, Z).

which has four variables W, X, Y and Z. The Index for p2 is set to zero since p2 is not a variable. Furthermore, the Arity is set to 2 since p2 has two parameters, W and X. As a clause is built into the knowledge base a single copy of each clause variable is stored in the clause Instance_Variables_Record, which is embedded in the Instance_Template field of the clause Head_Node. Subsequently, the template is copied to form a clause instance, each time the clause is matched with a goal, so that a single copy of a set of variable instances is used as the reference for all subgoals spawned by the matching process. This action is shown later in Figs 4.8 and 4.9.

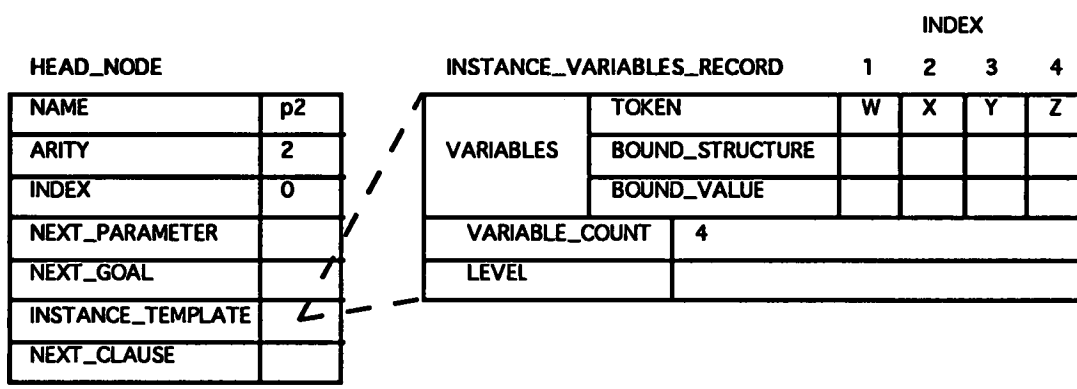

Fig 4.2 An exploded view of the logic Instance_Template.

2 The Arity of a predicate is equal to the number of its parameters.

Fig 4.3 shows a rule Sub_Goal_Node allocated to the variable X in the body of rule p2. The index position of the variable is recorded as 2 to match its position in the Instance_Template Instance_Variables_Record; this value is used during inferencing to locate the bound values of an instance of X.

SUB_GOAL_NODE

NAME	X
ARITY	0
INDEX	2
NEXT_PARAMETER	
NEXT_GOAL	

Fig 4.3 A logic rule Sub_Goal_Node allocated as a variable.

In each case, the Next_Parameter is used to point to the first parameter of each bracketed term, whereas the Next_Goal pointer locates the subgoal or element following an ':-' or ',' operator. In addition, a pointer is provided in the Head_Node for connection to the Next_Clause in the knowledge base, and a pointer is provided in the First_Sub_Goal node for connection to 'or' subgoals. Consequently, the presence of the ':-', ',' and ';' operators is not stored explicitly, since they are implied by the representation structure. For example, the fact

p1(s1(a, W), s2(X, b)).

is represented as shown in Fig 4.4

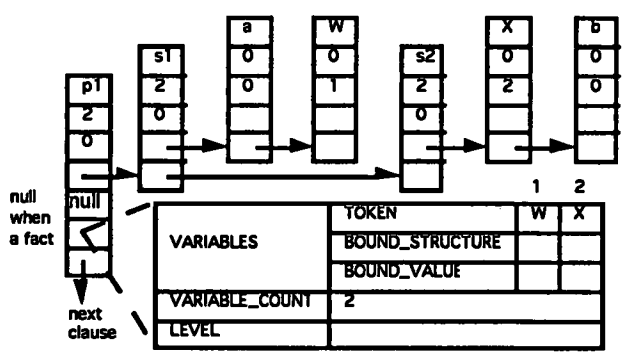

Fig 4.4 Representation of a logic fact.

and the rule

p2(W, X) :- p3(W, Y); p4(c, X, Y, Z).

is represented as shown in Fig 4.5.

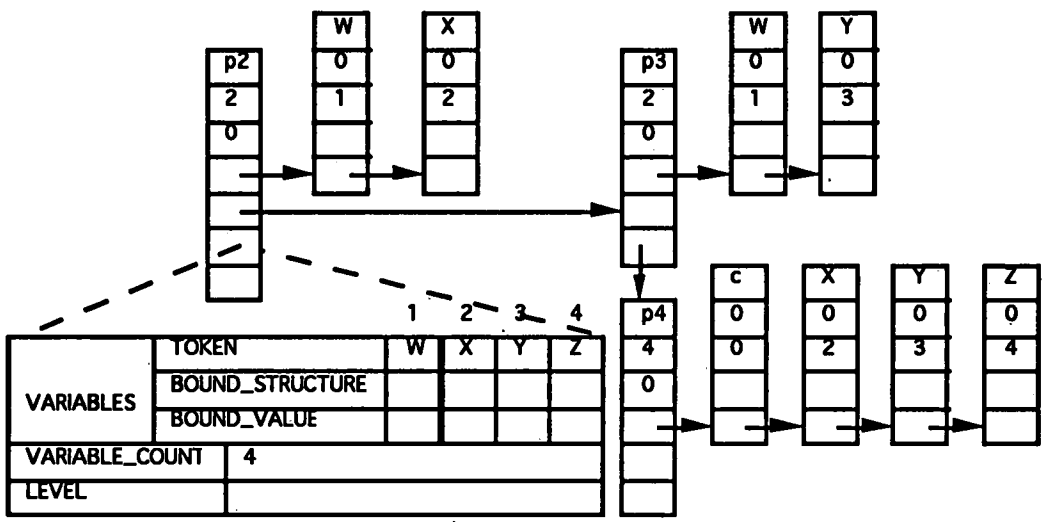

Fig 4.5 Representation of a logic rule.

The complete knowledge base structure that results from the PROLOG knowledge base, Bratco[22], given in Fig 2.1, is shown in Fig 4.6.

4.2.2 Queries

Syntactically, a query is the same as the clauses in the knowledge base. Consequently, a query is transformed in the same way. However, since the abstract knowledge type is designed to be embedded, the queries are determined by context and generated internally, rather than interactively as is the case in a normal PROLOG implementation.

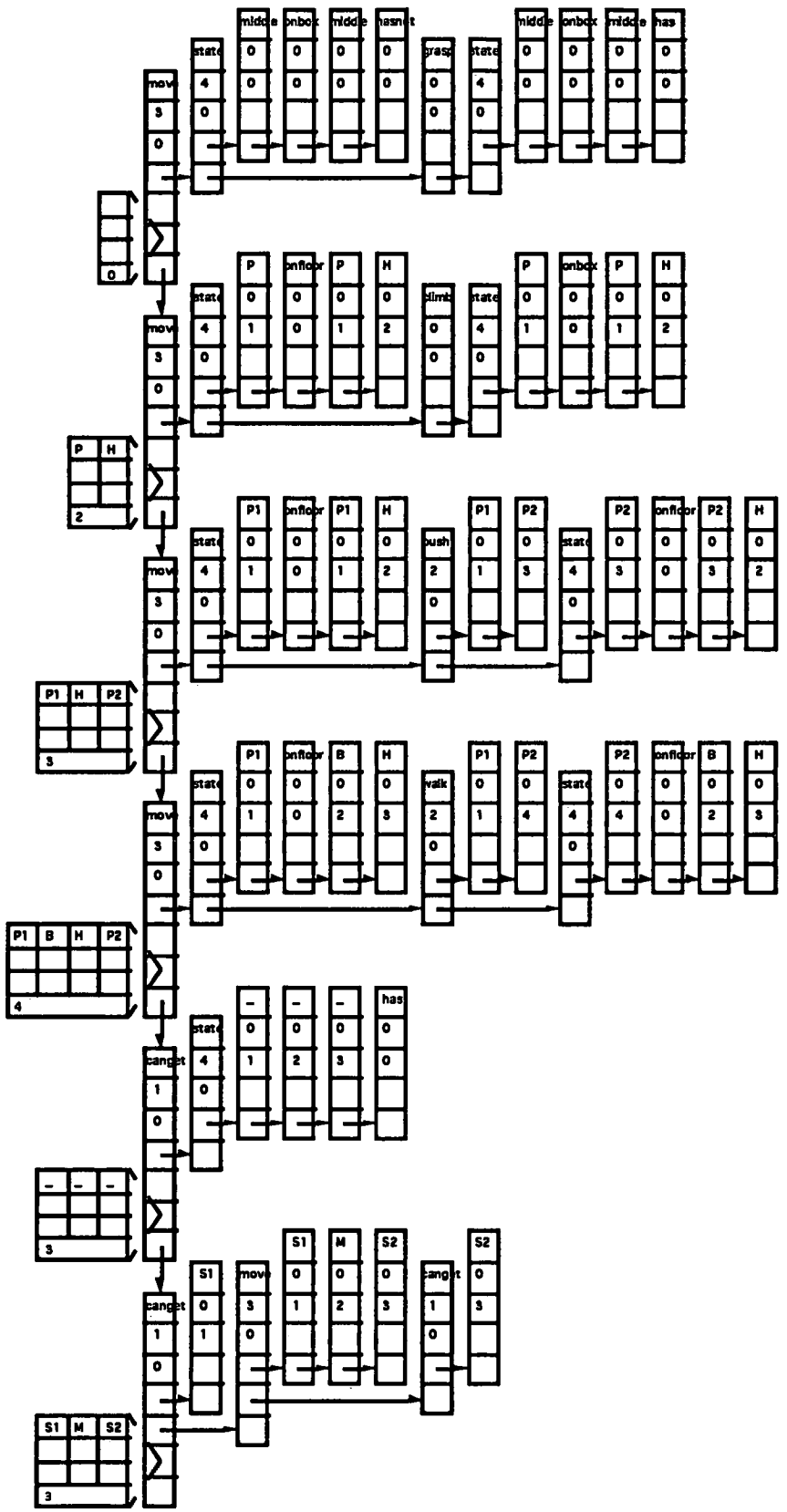

Fig 4.6 A complete logic knowledge base structure.

4.3 Knowledge Base Operations

4.3.1 Build

An internal Build operation is triggered automatically on instantiation of a generic logic abstract knowledge type. This operation transforms the external text file into the knowledge base architecture described in 4.2.1.

4.3.2 Resolution and Unification

Once the internal representations of the knowledge base and query have been built, the query subgoals are added to a list of Goals_To_Solve. The resolution process then recursively uses the first subgoal from the list of Goals_To_Solve and attempts to match the subgoal with the Head_Node of a clause in the knowledge base. As each subgoal is successfully matched against a clause in the knowledge base, a unique set of variables is created, by copying the clause Instance_Template Instance_Variables_Record, to represent the matched clause. The instance of the matched Instance_Template Instance_Variable_Record is linked to the appropriate matched clause structure in the knowledge base, using an instance of the record shown in Fig 4.7; this implements structure sharing.

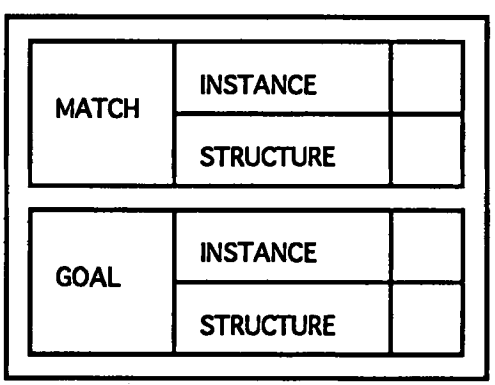

Fig 4.7 The logic goal-match record.

The goal-match record thus binds the instance variables for the goal and matched clauses to the appropriate knowledge base structure from which they were derived. This effect is shown in Fig 4.8 for the query canget(state(atdoor, onfloor, atwindow, hasnot)) on the knowledge base given in Fig 4.6, where the query is shown matching the rule canget(S1):- move(S1, M, S2), canget(S2). In this example, the query has no variables. The instance variables for the matching clause canget have been allocated and connected to the goal match record together with the matching clause structure.

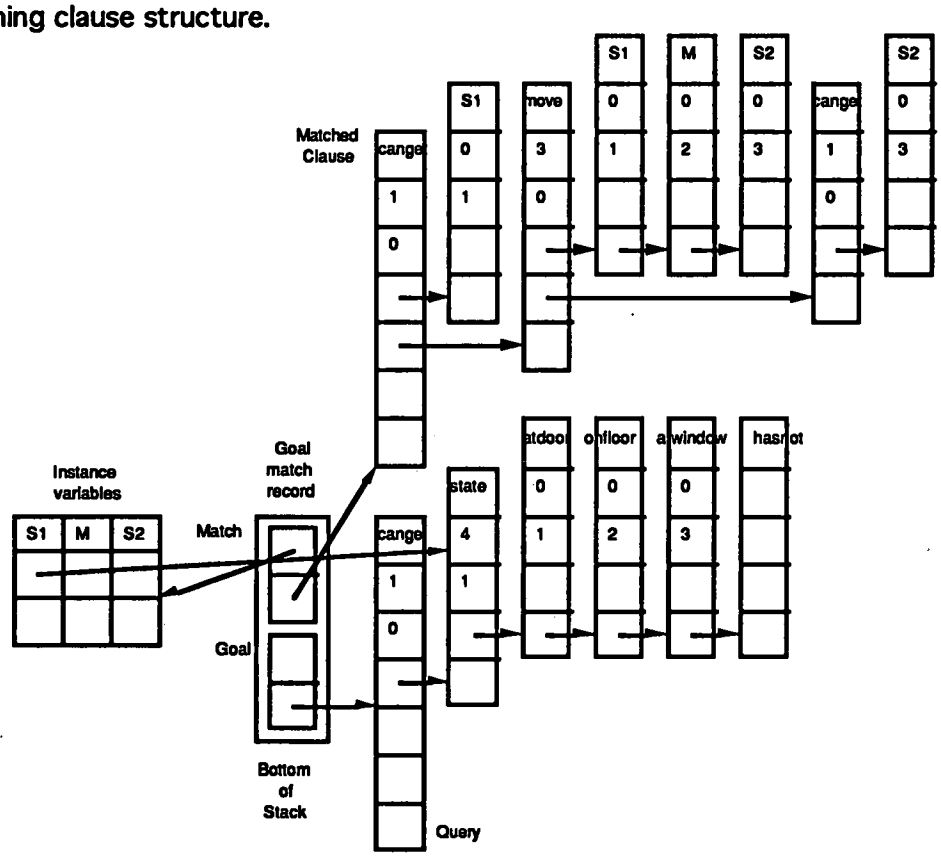

Fig 4.8 The logic query canget(state(canget(atdoor,onfloor,atwindow,hasnot)).

A successful match is followed by unification, where each component of the subgoal and matched Head_Node are compared. Atoms are checked for equality, and variables unified by pointing the variable instance at the variables and structure that form the substitution; in this case the variable S1 is shown instantiated to the state structure of the query. An unsuccessful unification results in these bindings being

undone, and the search for a match continued at the next clause lower in the knowledge base. However, if the unification is successful, then any subgoals associated with the matched Head_Node are added to the front of the list of Goals_To_Solve. The resolution unification process continues until all subgoals on the list of Goals_To_Solve have been satisfied, or until the search of the knowledge base has been exhausted. The next level in the dynamic structure, matching the subgoal move(S1, M, S2) with the fact move(state(P1, onfloor, B, H), walk,(P1, P2), state(P2, onfloor, B, H)), is shown in Fig 4.9.

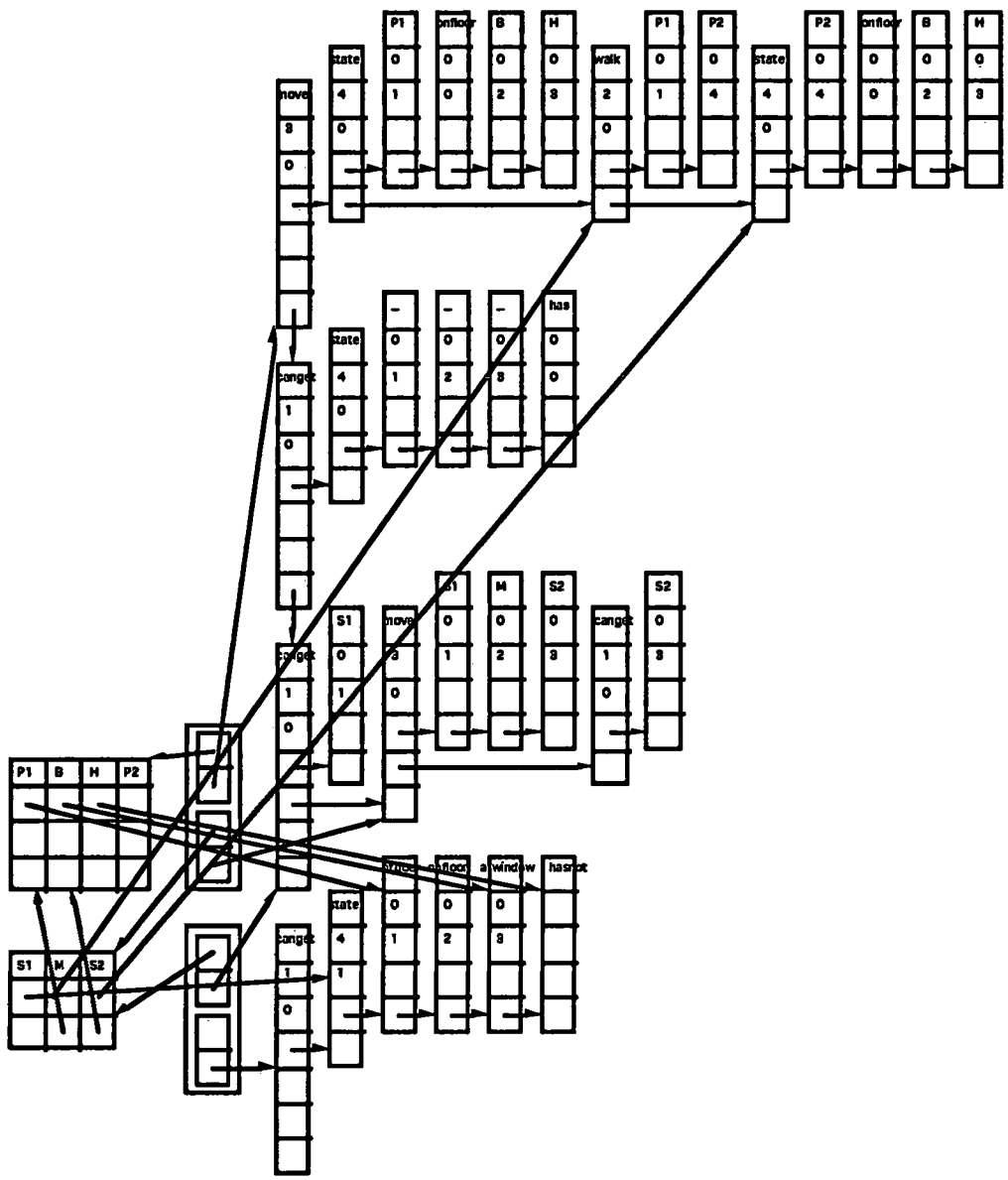

Fig 4.9 The match for the first subgoal move(S1, M, S2).

A second goal match record is created on the stack and the subgoal variables and structure pointers assigned. A match is found and the instance variables allocated. Unification of S1, which is already instantiated to state(atdoor, onfloor, atwindow, hasnot), with state(P1, onfloor, B, H) takes place by instantiating P1, B and H to atdoor, atwindow and hasnot respectively. In addition, the subgoal variables M and S2 are instantiated to the structures walk(P1, P2) and state(P2, onfloor, B, H) respectively. The variables associated with M and S2 are located by assignments into their Instance_Variables_Record to complete the unification. This dynamic process continues until the query is satisfied.

4.3.3 Backtracking

Backtracking is achieved by first undoing any bindings that have been made at the current stack level. Then, if the current subgoal is the first of a set of ',' subgoals attached to a particular clause, all associated ',' subgoals are deleted from the list of Goals_To_Solve. Furthermore, if there is an 'or' set of subgoals at this point, the current subgoal is reset to the first subgoal of the 'or' option, and the remaining 'or' subgoals added to the list of Goals_To_Solve. A search of the knowledge base is then resumed at the same recursive level for a match on the first 'or' subgoal. However, if there is no 'or' option at this point, a return is made to the subgoal at the previous recursive level, variables are unbound, and a search resumed at the next clause lower down in the knowledge base at which the previous match failed. Finally, if the failed subgoal is not a first subgoal, the failed subgoal is added to the front of Goals_To_Solve, a return is made to the previous subgoal at the next lower recursive level, variables are unbound, and a search of the knowledge base resumed on the subgoal at the old level. These effects are shown in Fig 4.10, Fig 4.11 and Fig 4.12 for a knowledge base containing the clauses

Q1 :- (G1, G2, G3); (G4, G5)., Q2. and the query Q1, Q2.

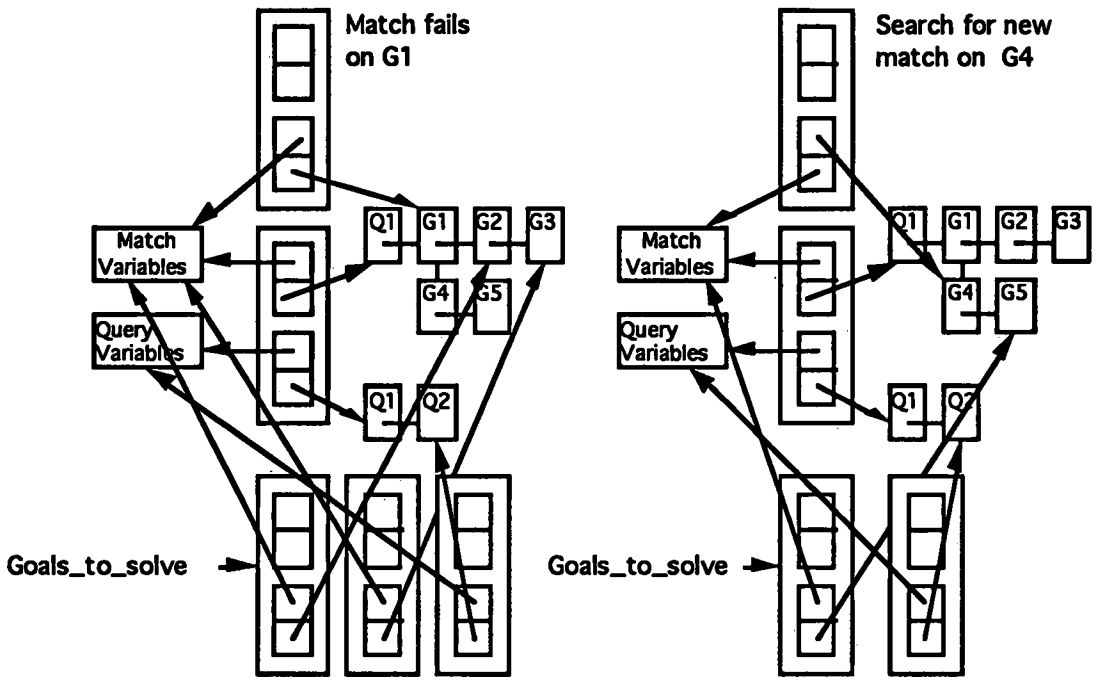

Fig 4.10 Backtracking from a subgoal with an 'or' option.

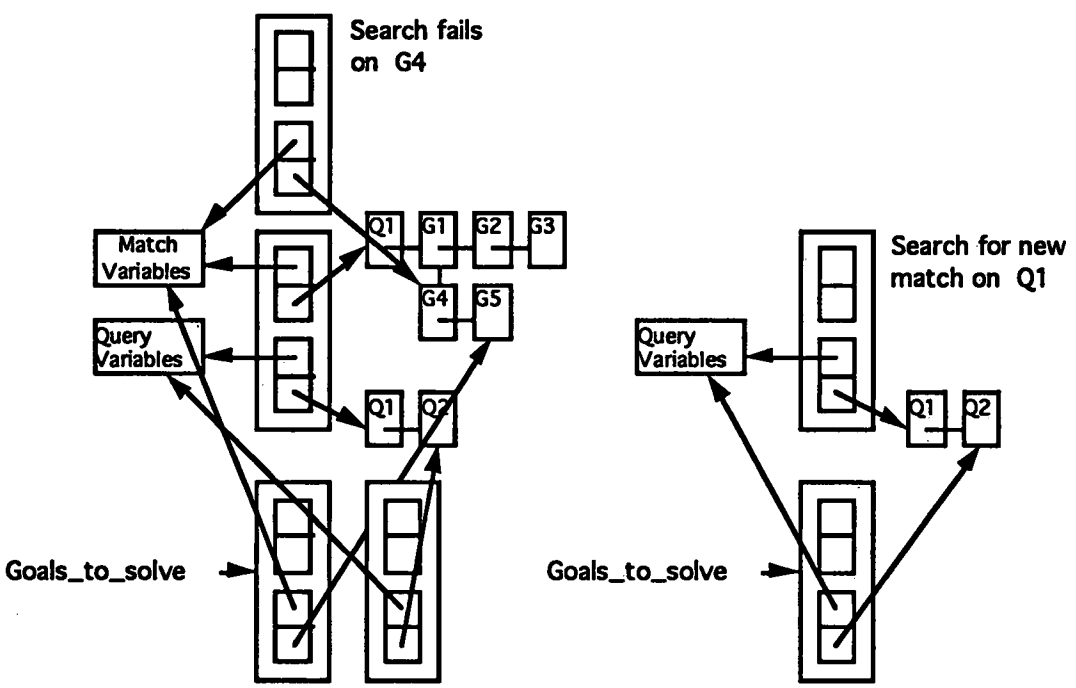

Fig 4.11 Backtracking from a first subgoal.

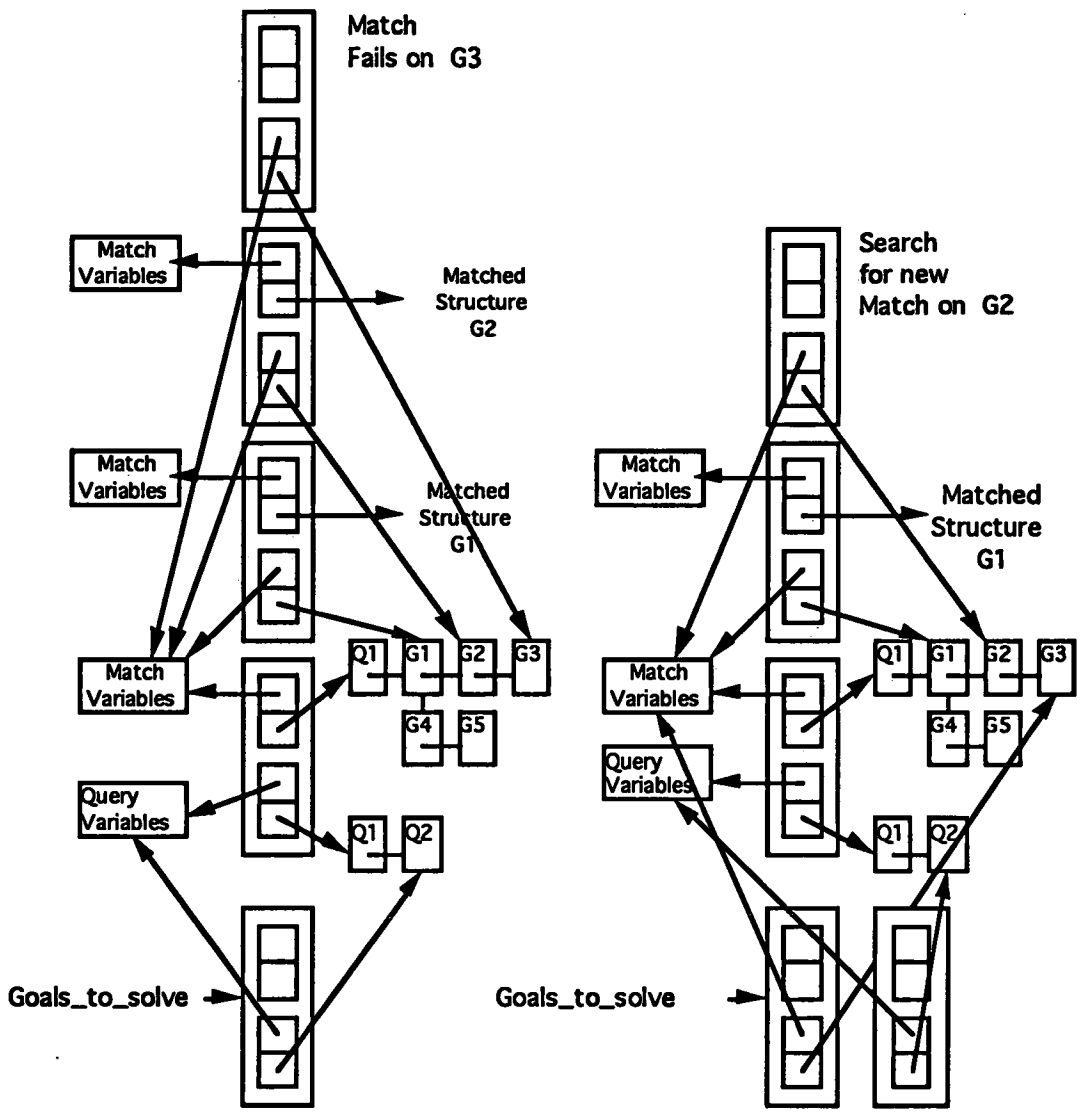

Fig 4.12 Backtracking from a subgoal.

4.3.4 Cut

As resolution proceeds, the level at which a set of instance variables come into existence is recorded in the associated instance of the Instance_Variables_Record. This value indicates the position of the instance variable's parent clause on the recursive stack. On backtracking past a cut, the level value, recorded in the Instance_Variables_Record, is used to unwind the goal-match stack until the parent recursive level is reached. At this point a further backtrack step is taken in order

to push the parent of the cut back onto Goals_To_Solve; this effectively prevents any further search on the parent clause of the cut. For example, the state of the goal-match stack prior to backtracking, for a knowledge base containing the clauses

A :- B, C, D.
B.
C :- P, !, Q, R.
P.

and the query A, is shown in Fig 4.13, and the result of backtracking shown in Fig 4.14.

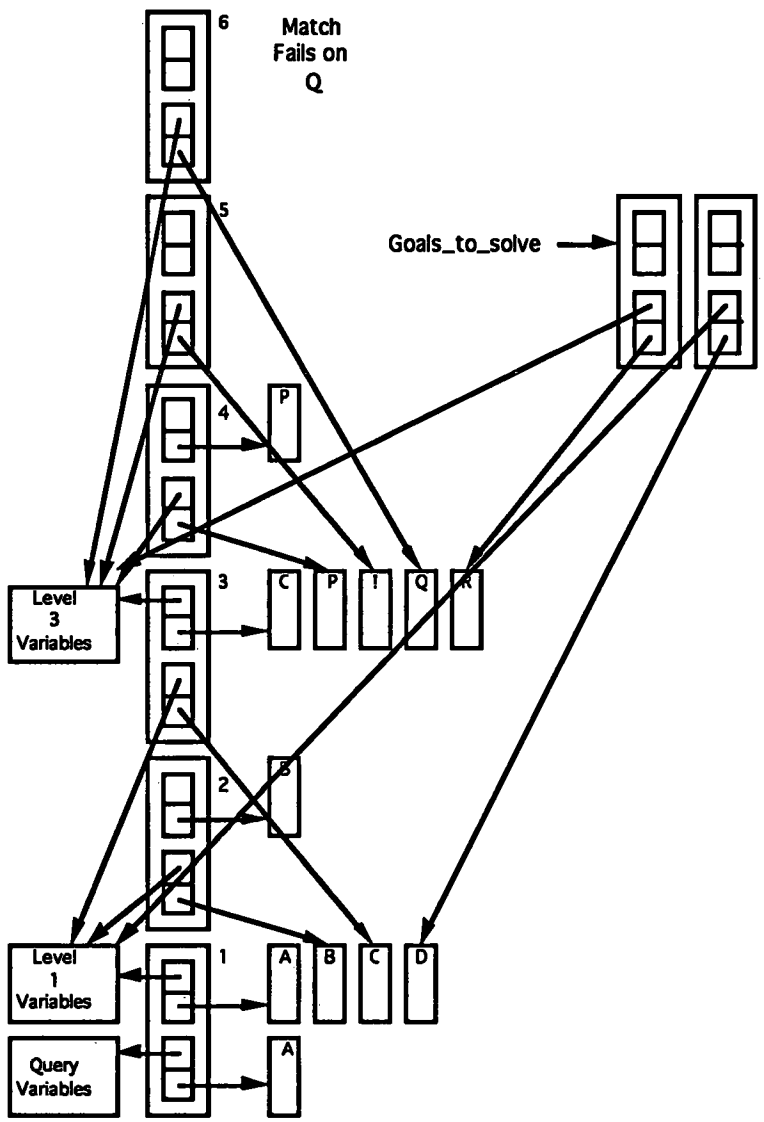

Fig 4.13 A failure after a cut.

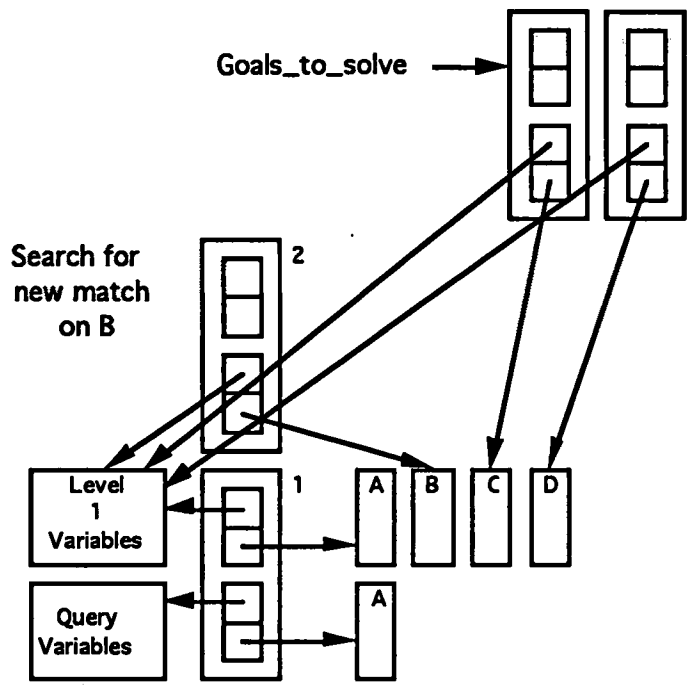

Fig 4.14 The result of backtracking past a cut.

4.3.5 Fail

On identifying a fail point, the system simply backtracks to the next lower recursive level and resumes the search of the knowledge base on the subgoal at that level.

4.3.6 Built-in-Operators

The ability to handle built-in operations is provided, although only a limited number of operations are implemented; these are 'is', '=' and '/='. For example, the infix representation of '/=' is first converted to prefix notation by the knowledge base Build algorithm. Subsequently, on detection of the operator during the inferencing process, the associated logical expression is evaluated by determining whether the operands can be unified.

4.4 Control

Since the implementation of the abstract knowledge type is generic, multiple independent instances can be instantiated. The question then arises, can these instances be made to co-operate in order to come to an agreed solution to an initial query? Such a set of co-operating knowledge-based components would be excellent for modelling situations where multiple experts are required to consult and co-operate when solving complex problems; for example, the situation assessment and resource allocation processes planned for the next generation command and control systems. What is needed is a means of suspending the inference process of one or more logic abstract knowledge type instances, while other instances are consulted. This process of suspension, consultation and re-consultation continues until an agreed solution is found or the knowledge bases are exhausted.

The Ada tasking model is an excellent mechanism for implementing this strategy. Each instance of the logic abstract knowledge type contains two independent tasks; an inference task to carry out the backward chaining, and a control task, which is responsible for maintaining the dialogue between the component and its environment. The logic abstract knowledge type architecture is shown in Fig 4.15.

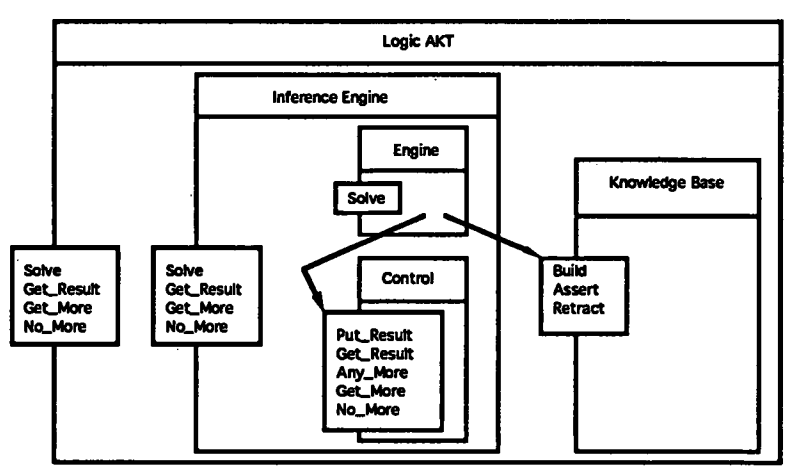

Fig 4.15 Logic abstract knowledge type architecture.

An external query triggers the engine to activate the Solve task. Subsequently a result is made available at the control interface, which is collected from the user environment; in the case of this research, from a knowledge source in which the instance is embedded.

Each control task has entry points for Put_Result, Get_Result, Any_More, Get_More and No_More; Get_Result, Get_More and No_More are called by the user component, and Put_Result and Any_More by the inference engine. Once the query has been asked of a particular logic instance, the user component uses the result as a means of consulting one or more other logic instances, while the original instance remains poised at the last solution. After consultation, if the solution is acceptable, the user component signals satisfaction through the No_More entry; on the other hand, if the result is unacceptable to the other logic instances, the user component asks for an alternative solution through the Get_More entry. This process continues until an agreed solution is reached or the knowledge base instances are exhausted. Fig 4.16 shows four co-operating abstract knowledge types where the user component has queried Logic_1 and Logic_2 and is currently getting confirmation from Logic_3. Logic_1 and Logic_2 are suspended at Any_More, while Logic_4 is awaiting consultation.

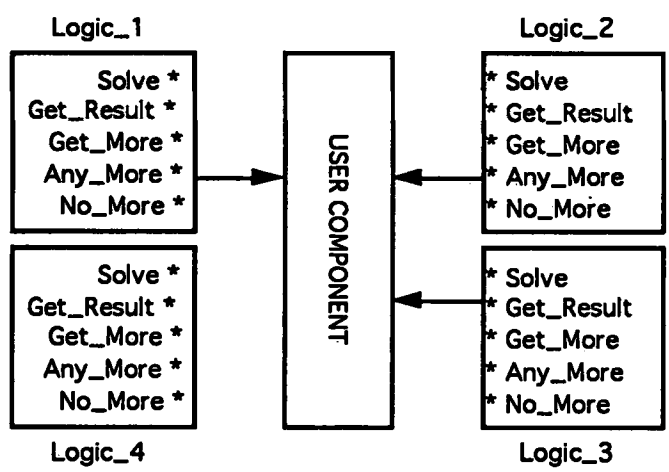

Fig 4.16 Co-operating logic AKTs.

4.5 Analysis

Although this research does not address the real-time issues of the abstract knowledge type implementations, it is necessary to analyse and test the components so that implementors of future components are aware of the characteristics associated with the components described in this thesis.

4.5.1 Ada Package Structure

The Ada package dependency structure is shown in Fig 4.17.

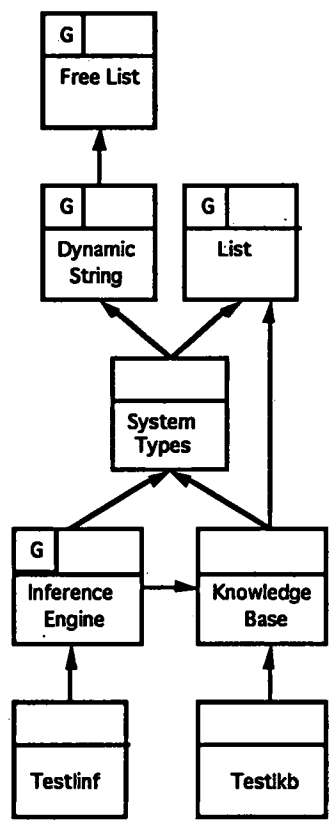

Fig 4.17 Logic Ada package dependencies.

The three generic packages, Free List, Dynamic String and List are widely used throughout the implementation, as is System Types. The core of the logic

abstraction is encapsulated in the Knowledge Base and Inference Engine packages. Test subprograms are built into both the Knowledge Base and Inference Engine; each is activated from simple test harnesses, Testlkb and Testlinf. The code listings are given in Annex A.

A user component instantiates the generic Inference Engine to establish an instance of the logic abstract knowledge type. In this case, the generic definition is very simple, comprising a single parameter representing the logic knowledge base name.

```
generic
      NAME : in STANDARD.STRING;
package INFERENCE_ENGINE is ...
```

However, this simple specification provides the capability to create multiple, unique and independent instances of the abstract knowledge type.

4.5.2 Knowledge Base Specification

The specification of Knowledge Base uses discriminants to control the size of an Instance_Variables_Record, and the structure of the Kb_Node_Record:

```
type NODE_TYPE is
(HEAD_NODE, FIRST_SUB_GOAL_NODE, SUB_GOAL_NODE, PARAMETER_NODE);
type KB_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE);
type KB_NODE_PTR_TYPE is access KB_NODE_RECORD;
subtype VARIABLE_RANGE is NATURAL range 0..40;
type INSTANCE_VARIABLES_RECORD(SIZE : VARIABLE_RANGE := 1);
type INSTANCE_VARIABLES_RECORD_PTR is
      access INSTANCE_VARIABLES_RECORD;

type VARIABLE_RECORD is
record
      TOKEN                :    SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.STRING;
      BOUND_STRUCTURE      :    KB_NODE_PTR_TYPE;
      BOUND_VALUE          :    INSTANCE_VARIABLES_RECORD_PTR;
end record;
```

```
type INSTANCES is array(POSITIVE range <>) of VARIABLE_RECORD;
type INSTANCE_VARIABLES_RECORD(SIZE : VARIABLE_RANGE := 1) is
record
      VARIABLES              :     INSTANCES(1 .. SIZE);
      VARIABLE_COUNT         :     NATURAL := 0;
      LEVEL                  :     NATURAL := 0;
end record;

type KB_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE) is
record
      NAME                   :     SYSTEM_TYPES_PACKAGE.
                                   DYNAMIC_STRING.STRING;
      ARITY                  :     NATURAL := 0;
      INDEX                  :     VARIABLE_RANGE := 0;
      NEXT_PARAMETER         :     KB_NODE_PTR_TYPE;
      NEXT_GOAL              :     KB_NODE_PTR_TYPE;
      case KIND is
            when HEAD_NODE                =>
                  CLAUSE                  :     SYSTEM_TYPES_PACKAGE.
                                                DYNAMIC_STRING.STRING;
                  INSTANCE_TEMPLATE       :     INSTANCE_VARIABLES_RECORD;
                  NEXT_CLAUSE             :     KB_NODE_PTR_TYPE;
            when FIRST_SUB_GOAL_NODE =>
                  OR_SUB_GOAL             :     KB_NODE_PTR_TYPE;
            when SUB_GOAL_NODE |
                  PARAMETER_NODE          =>    null;
      end case;
end record;

type KB_RECORD is
record
      FIRST       : KB_NODE_PTR_TYPE;
      LAST        : KB_NODE_PTR_TYPE;
end record;
```

4.5.3 Specification of Knowledge Base Operations

The knowledge base operations are specified as:

```
procedure BUILD(
KB                : in out      KB_RECORD;
FILE_NAME         : in          STANDARD.STRING);

procedure ASSERT(
IN_CLAUSE         : in          STANDARD.STRING;
KB                : in out      KB_RECORD;
AT_BACK_OF        : in          BOOLEAN := TRUE);
```

```
procedure RETRACT(
CLAUSE          : in        STANDARD.STRING;
KB              : in out    KB_RECORD);
```

4.5.4 Inference Engine Specification

The inference engine data structures are specified as:

```
type MATCHED_RECORD is
record
      INSTANCE          : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
      STRUCTURE         : LOGIC_KB.KB_NODE_PTR_TYPE;
end record;


type GOAL_RECORD is
record
      INSTANCE          : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
      STRUCTURE         : LOGIC_KB.KB_NODE_PTR_TYPE;
end record;

type INSTANCE_RECORD is
record
      MATCH             : MATCHED_RECORD;
      GOAL              : GOAL_RECORD;
end record;
type INSTANCE_RECORD_PTR is access INSTANCE_RECORD;
type BIND_RECORD is
record
      INDEX       : POSITIVE;
      INSTANCE    : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
end record;

type BIND_RECORD_PTR is access BIND_RECORD;
```

The Bind_Record is used to establish a list of current bindings to assist in backtracking.

4.5.5 Specification of Inference Engine Operations

The Inference Engine operations are specified as:

```
task ENGINE is
        entry SOLVE(
        THIS_QUERY              : in        LOGIC_KB.KB_RECORD;
        THIS_KB                 : in        LOGIC_KB.KB_RECORD);
end SOLVE;

task CONTROL is
        entry PUT_RESULT(
        IN_LIST                 : in        SYSTEM_TYPES_PACKAGE.
                                            DYNAMIC_STRING_LIST_PACKAGE.
                                            LIST_TYPE);
        entry GET_RESULT(
        OUT_LIST                :       out SYSTEM_TYPES_PACKAGE.
                                            DYNAMIC_STRING_LIST_PACKAGE.
                                            LIST_TYPE);
        entry ANY_MORE(
        MORE                    :       out BOOLEAN);
        entry GET_MORE;
        entry NO_MORE;
end CONTROL;
```

4.6 Results

The following tables record the lines of code, CPU time and the number of dynamic allocations against each module, whereas the charts record the internal dynamic analysis of each module. The results are discussed in Chapter 9.

4.6.1 Knowledge Base

Statistics were collected while building a knowledge base containing 2300 facts and 1000 rules. The results are shown in Table 4.1 and Chart 4.1 to Chart 4.4.

Item	Lines of Code	CPU Time %	Ada new
Free List	49	4.3	1453
Dynamic String	828	85.9	115325
List	520	0.3	13900
System Types	397	0	0
Knowledge Base	1822	9.4	88000
Inference Engine	2828	n/a	n/a
Testlkb	7	0	0
Testlinf	8	n/a	n/a
Min:Sec		6:14.45	

Table 4.1 Logic knowledge base test results.

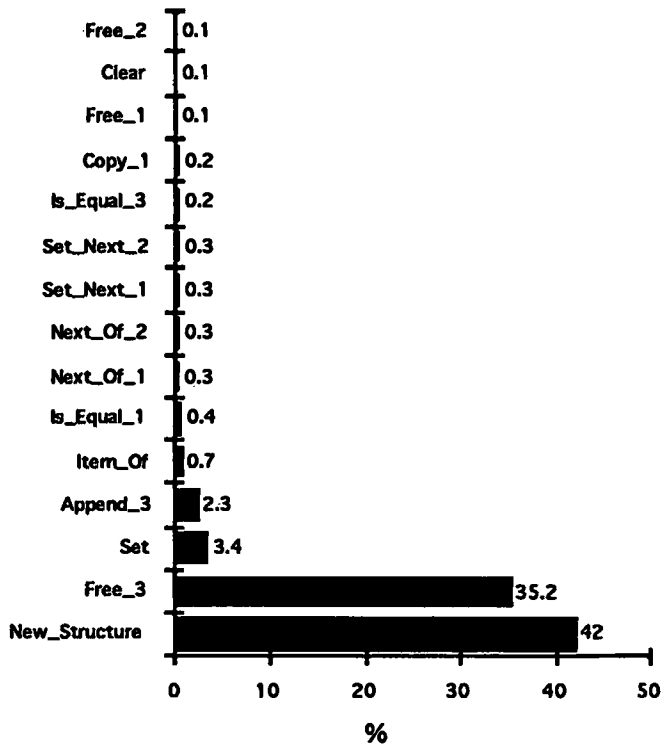

Chart 4.1 Logic knowledge base dynamic string analysis.

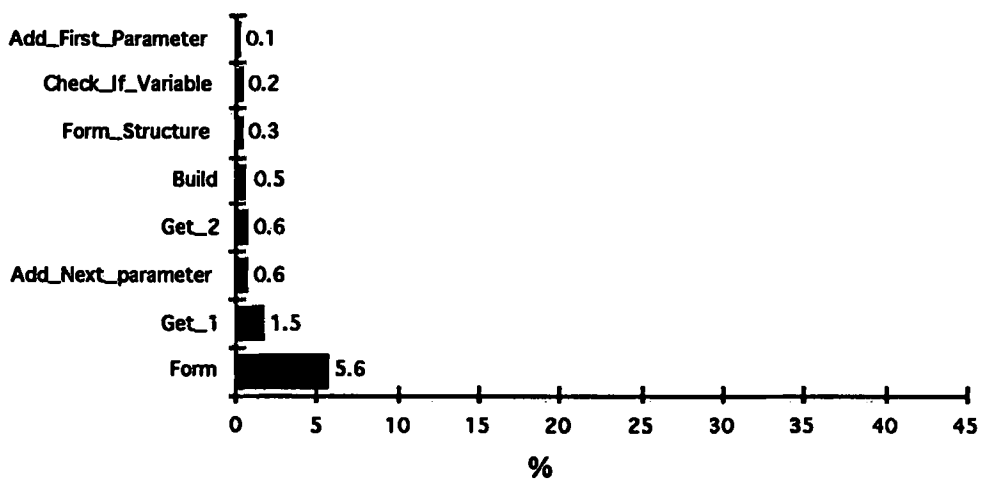

Chart 4.2 Logic knowledge base analysis.

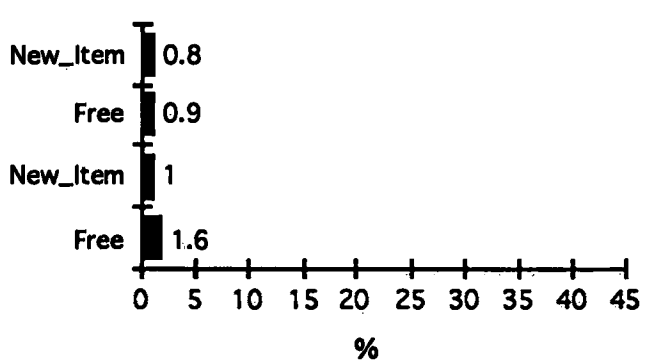

Chart 4.3 Logic knowledge base free list analysis.

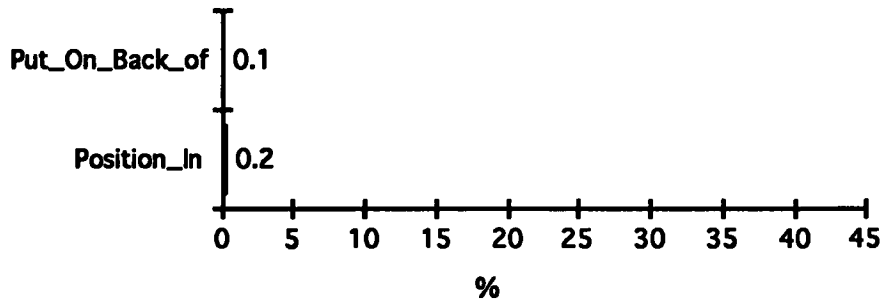

Chart 4.4 Logic knowledge base list analysis.

4.6.2 Inference Engine

Inference engine statistics were collected while repeatedly querying the inference engine 1000 times; the results are shown in Table 4.2 and Charts 4.5 to 4.9.

Item	CPU Time %	Ada new
Free List	1.2	1452
Dynamic String	71.0	156866
List	5.3	327139
System Types	0	0
Inference Engine	19.7	225000
Knowledge Base	1.1	15880
Testlkb	n/a	n/a
Testlinf	0.6	0
Min:Sec	2:56.09	

Table 4.2 Logic inference engine test results.

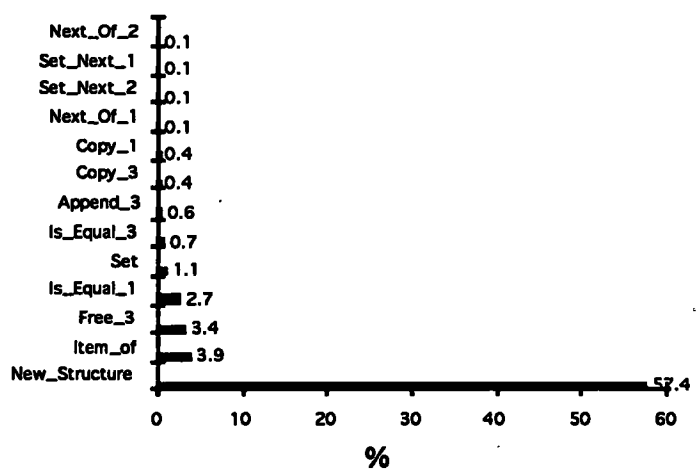

Chart 4.5 Logic inference engine dynamic string analysis.

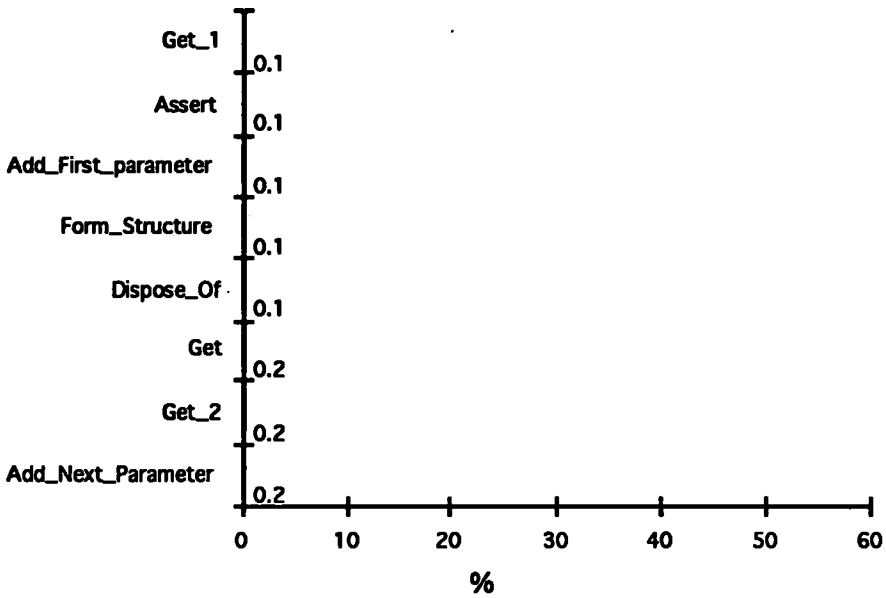

Chart 4.6 Logic inference engine knowledge base analysis.

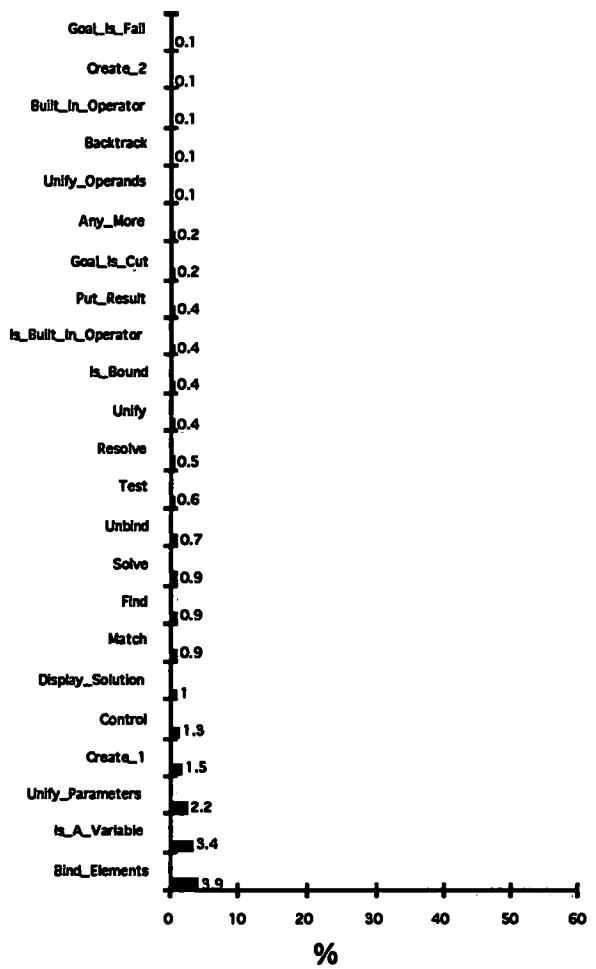

Chart 4.7 Logic inference engine analysis.

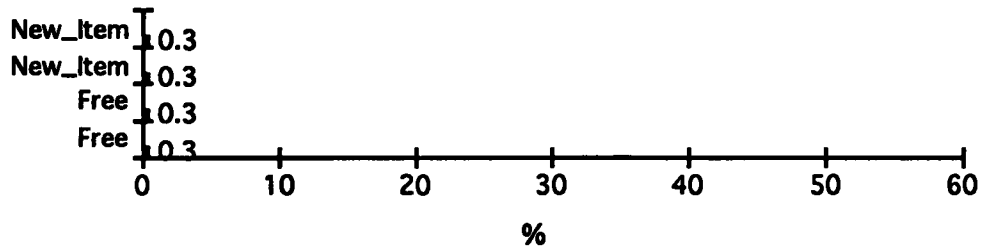

Chart 4.8 Logic inference engine free list analysis.

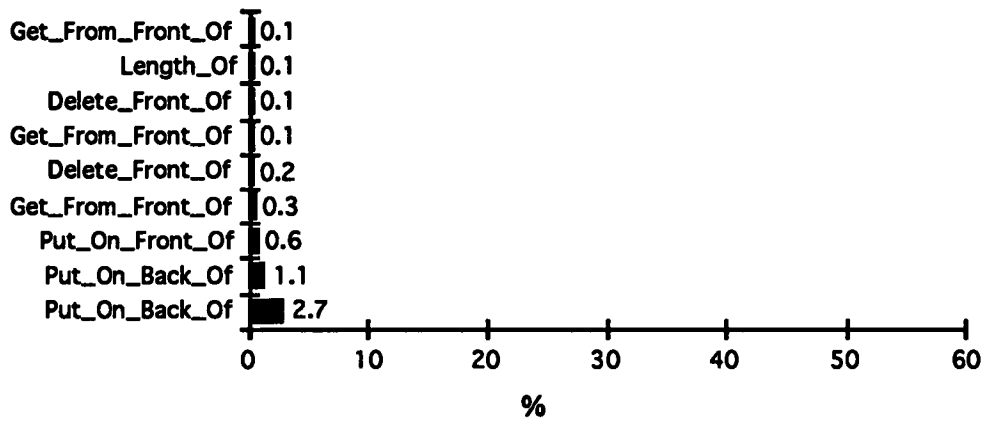

Chart 4.9 Logic inference engine list analysis.

4.7 Summary

This chapter describes the implementation of a generic logic abstract knowledge type. In particular, it describes the data structures and operations that are used to implement a restricted form of the PROLOG language. A novel control mechanism, that uses Ada tasks, is used to enable multiple instantiations of the abstract knowledge type in order to create instances that can co-operate to reach an agreed solution. This type of mechanism is excellent for modelling problem solutions that require multiple experts to co-operate in order to solve complex issues. The analysis identifies the Dynamic String abstract data type as consuming 85.9% of CPU time when building the knowledge base, and 71.0% when inferencing. The results of this experiment show that the use of the Dynamic String produces significant run-time overheads. Since logic processing is inherently dominated by

symbolic operations the overhead is to be expected. However, the size of the contribution is unacceptable in a real-time environment. One possible solution is to compile the knowledge base and operate the inference engine in the context of a symbol table.

Chapter 5

Implementation of a Rule Abstract Knowledge Type

5.1 Introduction

Chapter 2 identified rules as the most widely used knowledge representation in knowledge-based applications. The representation is easy to understand and appears a good model of how humans solve some of their problems. Furthermore, the representation and associated inference strategies are easy to automate. The aim of this chapter is to describe the implementation of a generic rule abstract knowledge type suitable for use with a blackboard architecture.

5.2 Rule Base Data Structures

The following simple rule syntax is used in the implementation:

IF <antecedent^1> AND <antecedent^2> AND ... AND <antecedent^n>
THEN <consequent>

Each of the antecedents and the consequent are represented by dynamic strings [16]. The number of rules and the number of antecedents is determined by the application user component. Although only one consequent is permitted in this implementation, the dynamic string representing the consequent is coded and subsequently interpreted by the application to produce multiple actions. An example of a rule

taken from the experiment described in Chapter 8 is

IF E101(L) AND I111(L)
THEN make_E101_and_I111_common

which shows that if the event E101(L), a request for a lecture period on module E101, and the event I111(L), a request for a lecture period on module I111, have occurred, they should share a common period.

5.2.1 Rule Base Architecture

A rule is constructed from the three nodes shown in Fig 5.1 and assembled as shown in Fig 5.2. The Fired field is used to signify that a rule has already been actioned and that it need not be considered again until cleared.

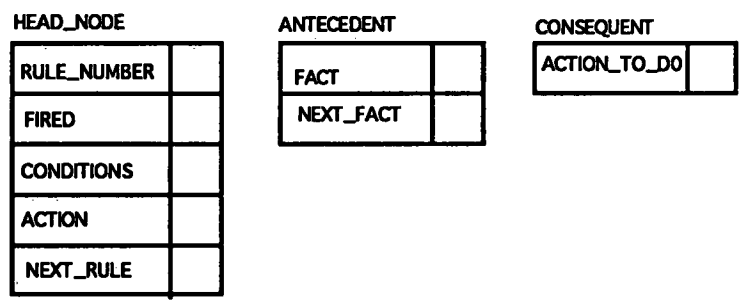

Fig 5.1 Rule base nodes.

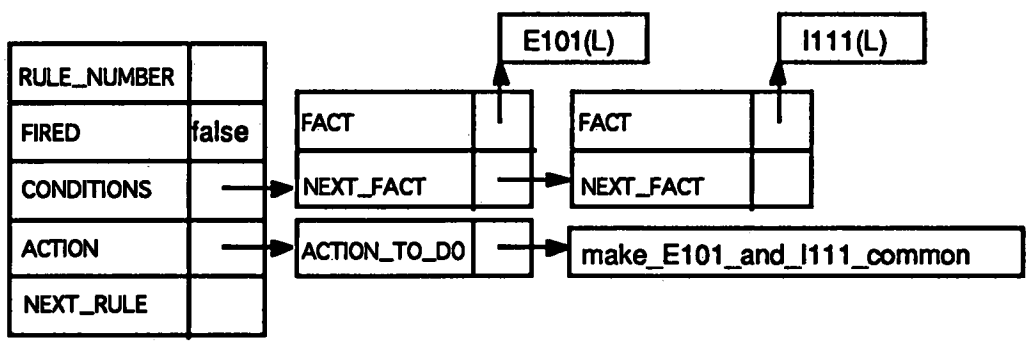

Fig 5.2 Rule structure.

A rule base is a collection of rules connected as shown in Fig 5.3. Since a user component may wish to order the rules to suit a particular task, a linear structure is used to support the rule base.

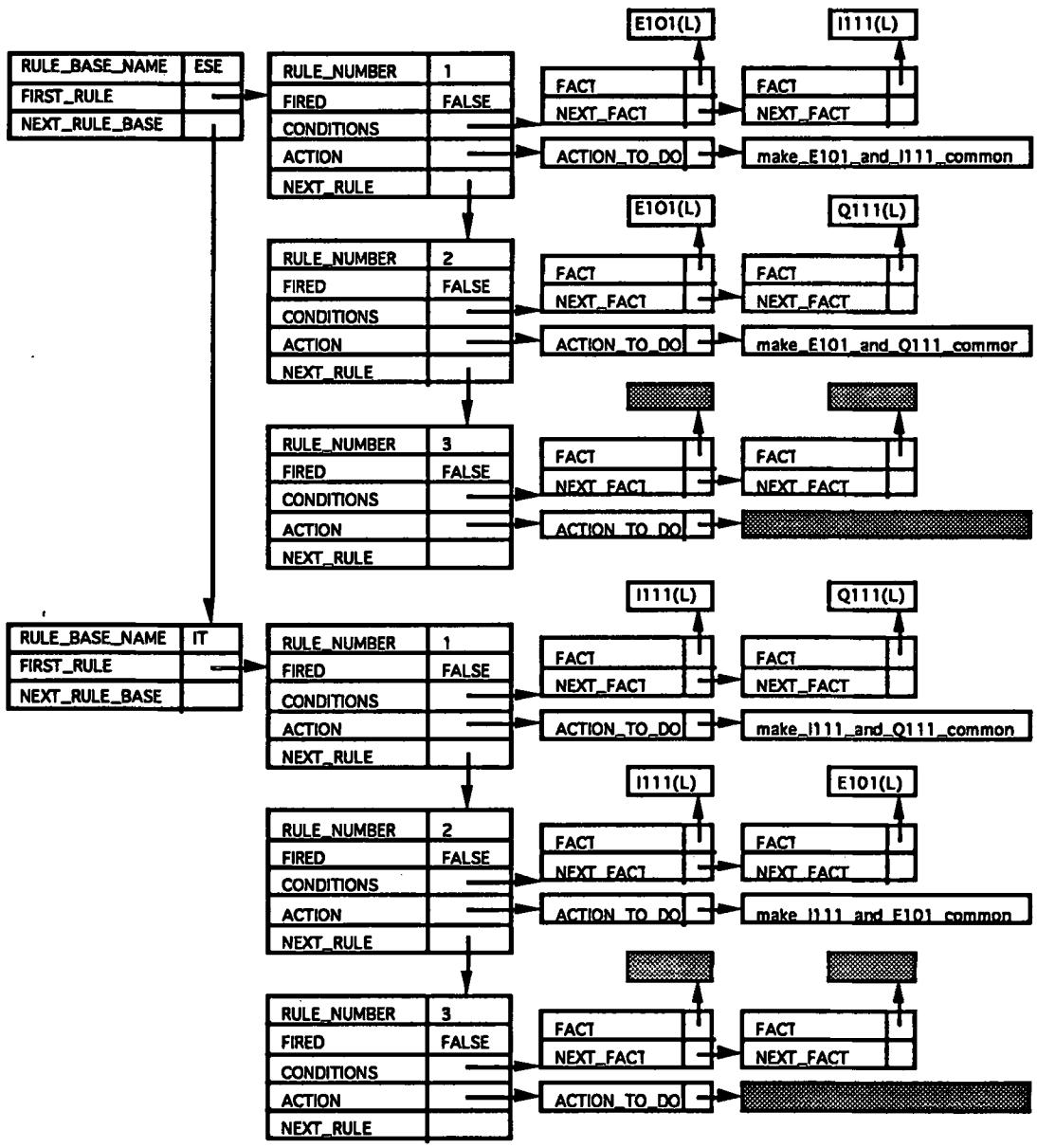

Fig 5.3 A partitioned rule base structure.

In anticipation of large rule bases, the rules are partitioned. The event causing a rule inference request, a change in the fact base, is used to determine the rule base partition to be searched. Since a linear search is used, partitioning reduces search

time. In addition, techniques such as the Rete algorithm [60] and rule compilation can be used to improve search efficiency.

The ordering of facts is not significant, but there is a need to minimise search times. Consequently, a balanced binary search tree is used to represent the fact base structure. Each fact is a pointer to an entry on the blackboard. Since the blackboard entries are determined by the application, the fact representation is generic. The generic blackboard entries, which form the rule base facts for this experiment, are discussed in Chapter 8. A fragment structure, comprising two fact bases from different instances of the rule abstract knowledge type, is given in Fig 5.4, which shows the two fact bases sharing entries on a common blackboard.

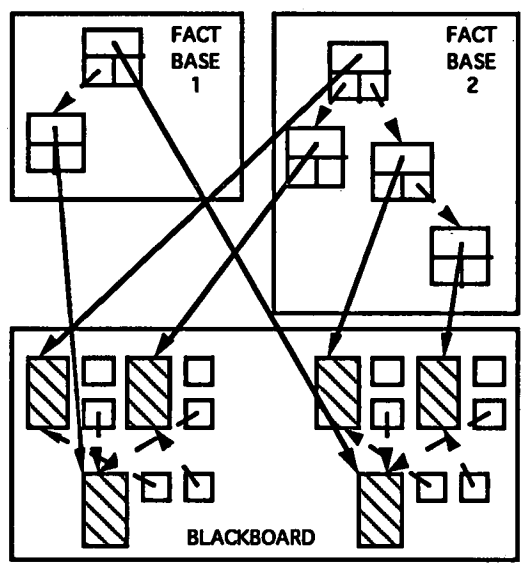

Fig 5.4 Fact base structure.

5.3 Rule Base Operations

The operations associated with the rule abstract knowledge type are much simpler than those needed to implement the logic abstract knowledge type. The rule abstract knowledge type architecture is shown in Fig 5.5.

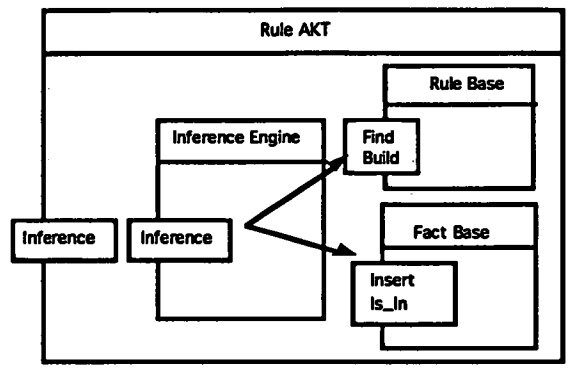

Fig 5.5 Rule AKT architecture.

5.3.1 Build

An internal Build operation, which transforms the external file representation of the rule base into the internal form shown in Fig 5.3, is triggered automatically on instantiation of a rule base generic component, and the rule base built during package elaboration. The Build process is shown in Fig 5.6.

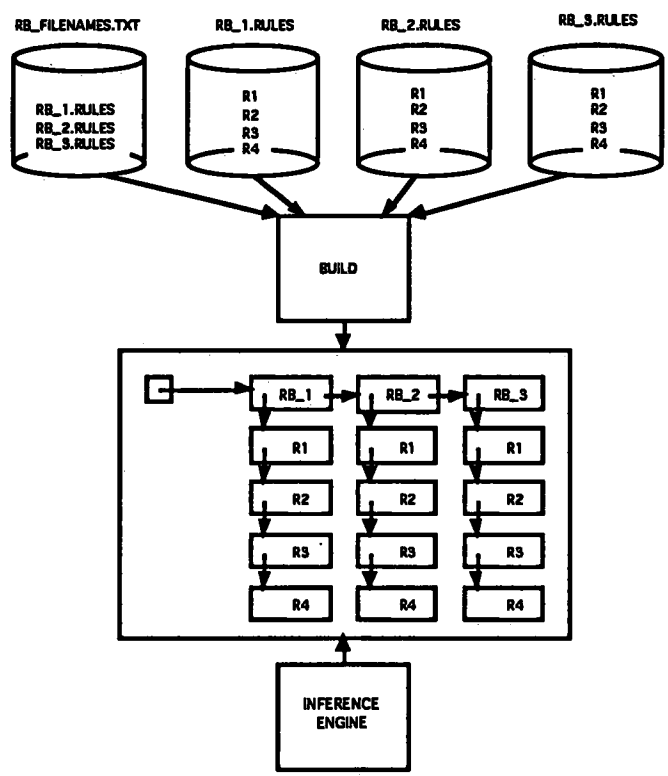

Fig 5.6 Rule base build process.

5.3.2 Inference

The forward chaining inference operation is activated by an event occurring on the blackboard; a change in the fact base. The activating event is used to select the appropriate rule base partition to search. As the search progresses, the antecedents of each rule are matched against the entries in the fact base. When all antecedents of a rule match entries in the fact base the rule is added to an agenda, a linear list; the agenda is returned to the activating component on completion of the inference cycle.

The agenda of triggered rules is used by the component making the inference request to modify the blackboard fact base. The consequent of each triggered rule is first decoded, followed by the user component taking appropriate action. Rule firing results in the addition or amendment of an entry on the blackboard so changing the set of recorded facts. In a conventional rule-based environment a change in the fact base causes further match and fire cycles as the system forward chains through the set of rules. In this case, where multiple rule bases share a global fact base, a change in the fact base triggers a different rule base inference engine which is monitoring another area of the blackboard. Consequently, this process is slightly different from what happens in a single rule-based system, in that forward chaining progresses over the blackboard, from rule base to rule base, rather than being confined within a single rule-based inference process.

5.4 Analysis

5.4.1 Ada Package Structure

The Ada package dependencies are shown in Fig 5.7.

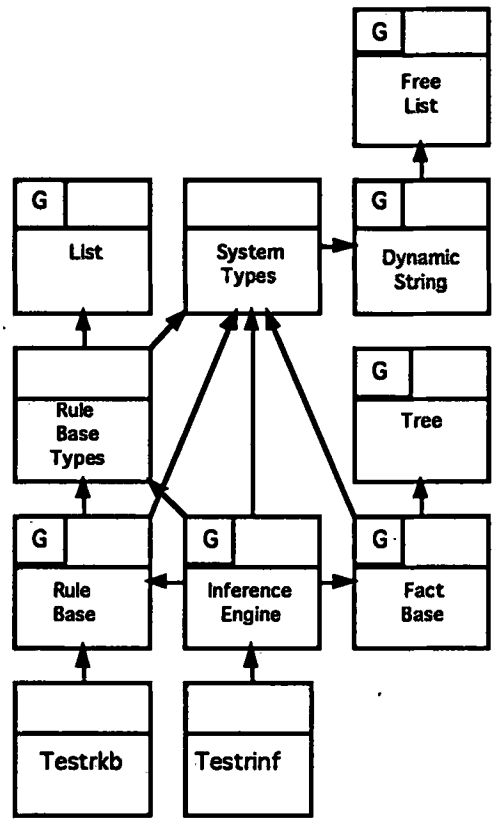

Fig 5.7 Rule abstract knowledge type Ada package dependencies.

As can be seen, the generic packages Free List, Dynamic String and List have been reused in this component, as is System Types. In addition, a generic Tree is provided to form the structure of the fact base. Two simple test harnesses, Testrkb and Testrinf, are provided to support component testing. The code listings are given in Annex A.

A user component instantiates the Inference Engine to establish an instance of the rule abstract knowledge type. In this case, the generic definition is more complex than that of the logic abstract knowledge type. The generic parameters to the Inference Engine specification, given below, are used to instantiate an instance of the generic Tree in order to form the fact base structure. However, this relatively simple specification provides the ability to create multiple, unique and independent instances of the abstract knowledge type.

```
with  SYSTEM_TYPES_PACKAGE,
      RULE_BASE_TYPES_PACKAGE,
      GENERIC_RULE_BASE_PACKAGE,
      GENERIC_FACT_BASE_PACKAGE;
generic
      RULE_BASES_FILENAME : STANDARD.STRING;
      type FACT_COMPOSITE_TYPE is private;
      type FACT_PTR_TYPE is access FACT_COMPOSITE_TYPE;

      with function "<"(
      LEFT        : in FACT_PTR_TYPE;
      RIGHT       : In FACT_PTR_TYPE) return BOOLEAN;

      with function ">"(
      LEFT        : in FACT_PTR_TYPE;
      RIGHT       : in FACT_PTR_TYPE) return BOOLEAN;

      with function IS_EQUAL(
      FACT        : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
      FACT_PTR    : in FACT_PTR_TYPE) return BOOLEAN;

      with function IS_LESS_THAN(
      FACT        : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
      FACT_PTR    : in FACT_PTR_TYPE) return BOOLEAN;

      with function IS_GREATER_THAN(
      FACT        : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
      FACT_PTR    : in FACT_PTR_TYPE) return BOOLEAN;

      with procedure PUT(
      FACT : in FACT_PTR_TYPE);
package GENERIC_RULE_BASE_INFERENCE_PACKAGE is...
```

5.4.2 Rule Base Types Specification

A discriminant is used to establish the structure of the rule base nodes. The specification is:

```
type NODE_TYPE is (HEAD_NODE, ANTECEDENT, CONSEQUENT);
type RULE_BASE_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE);
type RULE_BASE_NODE_PTR_TYPE is access RULE_BASE_NODE_RECORD;
type RULE_BASE_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE) is
record
      case KIND is
      when HEAD_NODE        =>
            RULE_NUMBER     : POSITIVE;
            FIRED           : BOOLEAN;
            CONDITIONS      : RULE_BASE_NODE_PTR_TYPE;
            ACTION          : RULE_BASE_NODE_PTR_TYPE;
```

```
            NEXT_RULE       : RULE_BASE_NODE_PTR_TYPE;
        when ANTECEDENT     =>
            FACT            : SYSTEM_TYPES_PACKAGE.
                                  DYNAMIC_STRING.STRING;
            NEXT_FACT       : RULE_BASE_NODE_PTR_TYPE;
        when CONSEQUENT     =>
            ACTION_TO_DO    : SYSTEM_TYPES_PACKAGE.
                                  DYNAMIC_STRING.STRING;
        end case;
    end record;

    type RULE_BASE_RECORD;
    type RULE_BASE_RECORD_PTR_TYPE is access RULE_BASE_RECORD;
    type RULE_BASE_RECORD is
    record
        RULE_BASE_NAME: SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FIRST_RULE: RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;
        NEXT_RULE_BASE : RULE_BASE_RECORD_PTR_TYPE;
    end record;

    package AGENDA_LIST_PACKAGE is new GENERIC_LIST_PACKAGE(
    ITEM_TYPE  => RULE_BASE_NODE_PTR_TYPE,
    IS_EQUAL   => IS_EQUAL);
```

5.4.3 Rule Base Specification

The following generic specification is used to establish an instance of Rule Base:

```
generic
    RULE_BASES_FILENAME : in STANDARD.STRING;
package GENERIC_RULE_BASE_PACKAGE is ...
```

5.4.4 Rule Base Operations

Two operations are provided in Rule Base:

```
procedure BUILD(
RULE_BASES_PTR              : in   out  RULE_BASE_TYPES_PACKAGE.
                                        RULE_BASE_RECORD_PTR_TYPE;
RULE_BASES_FILENAME         : in        STANDARD.STRING);

procedure FIND(
RULE_BASE                   : in        STANDARD.STRING;
RULE_PTR                    :      out  RULE_BASE_TYPES_PACKAGE.
                                        RULE_BASE_NODE_PTR_TYPE);
```

5.4.5 Fact Base Specification

The generic specification of Fact Base provides the detail to instantiate Fact Tree:

```
with  GENERIC_TREE_PACKAGE,
      SYSTEM_TYPES_PACKAGE,
      TEXT_IO;
generic
      type FACT_COMPOSITE_TYPE is private;
      type FACT_PTR_TYPE is access FACT_COMPOSITE_TYPE;
      with function "<"(
      LEFT  : in FACT_PTR_TYPE;
      RIGHT       : in FACT_PTR_TYPE) return BOOLEAN;
      with function ">"(
      LEFT  : in FACT_PTR_TYPE;
      RIGHT             : in FACT_PTR_TYPE) return BOOLEAN;
      with function IS_EQUAL(
      FACT              : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
      FACT_PTR          : in FACT_PTR_TYPE) return BOOLEAN;
      with function IS_LESS_THAN(
      FACT              : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
      FACT_PTR          : in FACT_PTR_TYPE) return BOOLEAN;
      with function IS_GREATER_THAN(
      FACT              : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
      FACT_PTR          : in FACT_PTR_TYPE) return BOOLEAN;
      with procedure PUT(
      FACT              : in FACT_PTR_TYPE);

package GENERIC_FACT_BASE_PACKAGE is
      package FACT_TREE_PACKAGE is new GENERIC_TREE_PACKAGE(
      ITEM_BASE_TYPE              =>    SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.STRING,
      ITEM_COMPOSITE_TYPE         =>    FACT_COMPOSITE_TYPE,
      ITEM_PTR_TYPE               =>    FACT_PTR_TYPE,
      "<"                         =>    "<",
      ">"                         =>    ">",
      IS_EQUAL                    =>    IS_EQUAL,
      IS_LESS_THAN                =>    IS_LESS_THAN,
      IS_GREATER_THAN             =>    IS_GREATER_THAN,
      PUT                         =>    PUT);

type FACT_BASE_RECORD is
record
      FACTS : FACT_TREE_PACKAGE.TREE_PTR_TYPE;
end record;
```

5.4.6 Specification of Fact Base Operations

Two operations are provided with Fact Base:

```
procedure INSERT(
FACT            : in        FACT_PTR_TYPE;
FACT_BASE       : in out    FACT_BASE_RECORD);

function IS_IN(
FACT            : in        SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.STRING;
FACT_BASE       : in        FACT_BASE_RECORD) return BOOLEAN;
```

5.4.7 Inference Engine Specification

No further data definitions are needed for the specification of Inference Engine.

5.4.8 Specification of Inference Engine Operation

Only one operation is needed in Inference Engine:

```
procedure INFERENCE(
RULE_BASE       : in        STANDARD.STRING;
FACT_PTR        : in        FACT_PTR_TYPE;
AGENDA          : in out    RULE_BASE_TYPES_PACKAGE.
                            AGENDA_LIST_PACKAGE.LIST_TYPE);
```

5.5 Results

The test results are recorded in the following tables and charts, but the results are discussed in Chapter 9.

5.5.1 Rule Base

A partitioned rule base containing 3700 rules was generated to provide the data for the Build test. The combined CPU times for each module are shown in Table 5.1 and the internal time distributions in Charts 5.1 to 5.4.

Item	Lines of Code	CPU Time %	Ada new
Free List	49	10.5	112
Dynamic_String	828	64.3	20454
System Types	397	16.7	0
List	520	0	0
Tree	526	n/a	n/a
Rule Base Types	136	0	0
Rule Base	498	8.0	20300
Fact Base	163	n/a	n/a
Inference Engine	229	n/a	n/a
Testrkb	10	0.1	0
Testrinf	84	n/a	n/a
Min:Sec		0:42.58	

Table 5.1 Rule knowledge base test results.

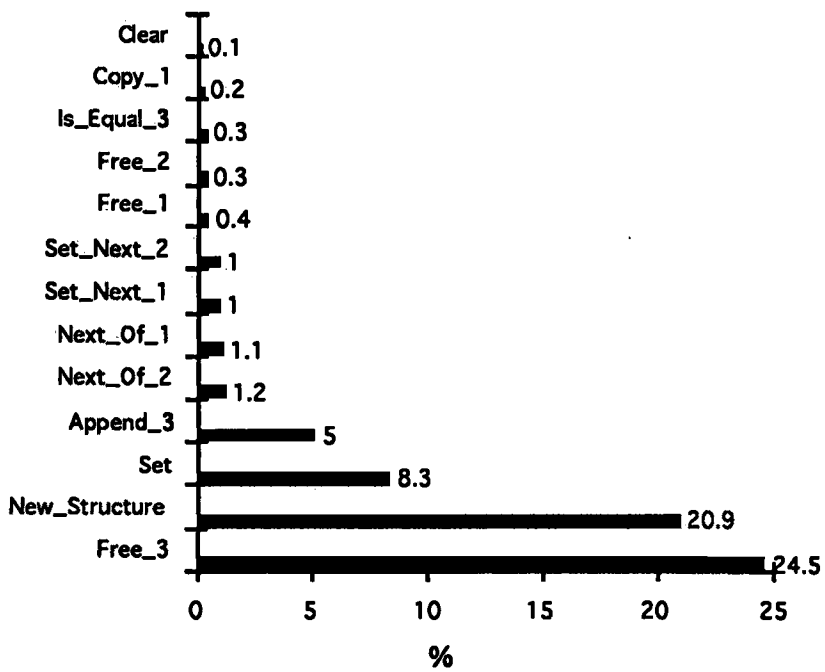

Chart 5.1 Rule knowledge base dynamic string analysis.

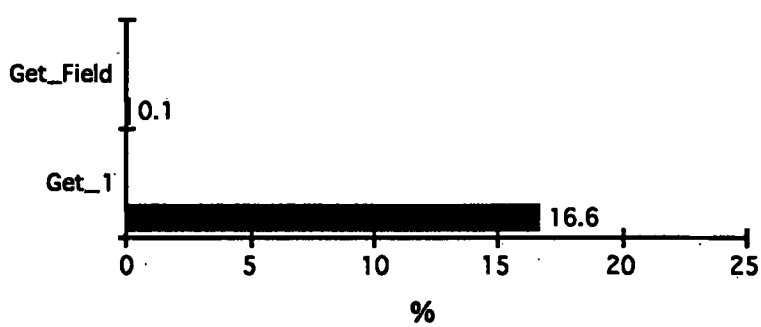

Chart 5.2 Rule knowledge base system types analysis.

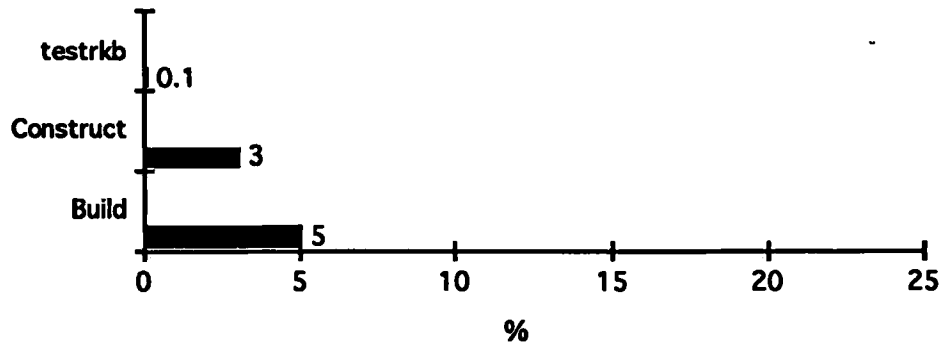

Chart 5.3 Rule base analysis.

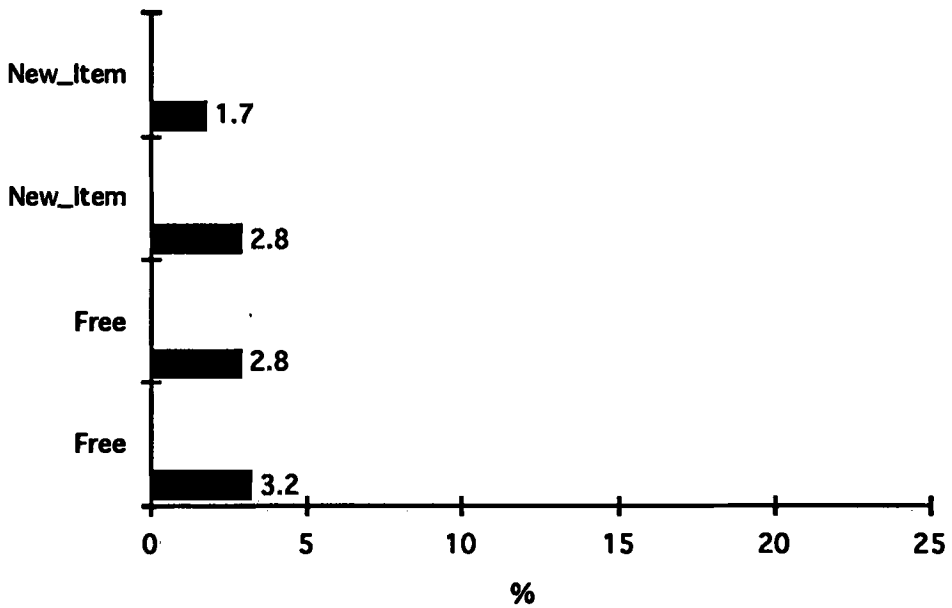

Chart 5.4 Rule knowledge base free list analysis.

5.5.2 Inference Engine

Inference Engine statistics were collected by repeatedly querying the Rule Base component. A test unit was developed which involved the construction of the Rule Base over which 1,000 inferences were made. The results of the inference test are shown in Table 5.2. and Charts 5.5 to 5.13.

Item	CPU Time %	Ada new
Free List	4.9	112
Dynamic String	47.8	258
System Types	12.4	0
List	9.6	1000
Tree	1.0	1000
Rule Base Types	0	0
Rule Base	6.8	203
Fact Base	1.0	0
Inference Engine	8.6	0
Testrkb	n/a	n/a
Testrinf	8.6	0
Min:Sec	0:00.52	

Table 5.2 Rule inference engine test results.

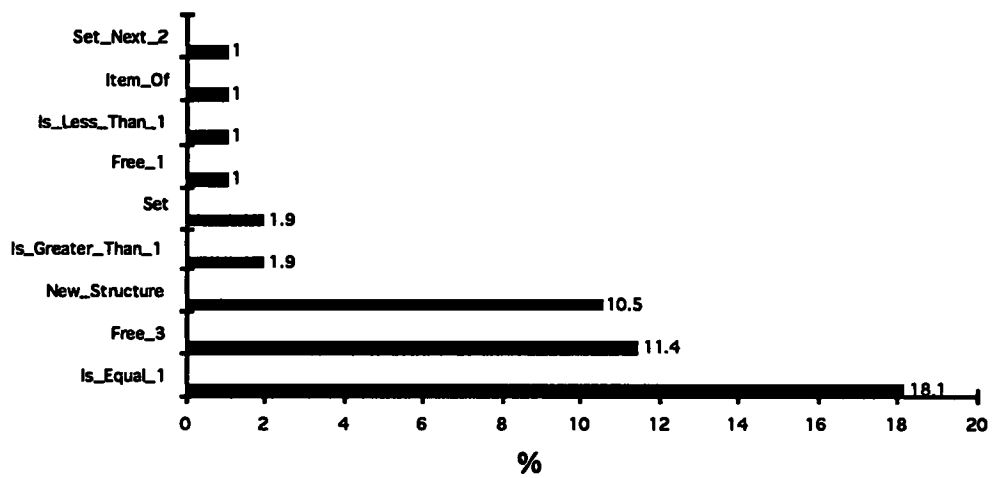

Chart 5.5 Rule inference engine dynamic string analysis.

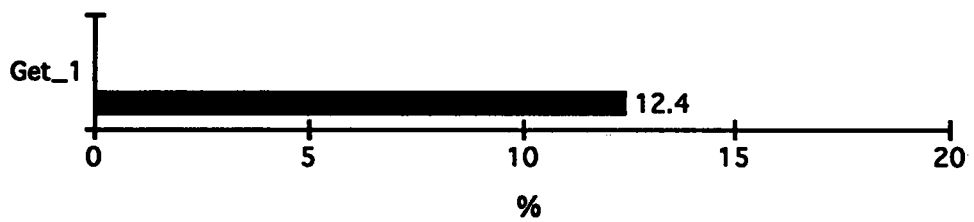

Chart 5.6 Rule inference engine system types analysis.

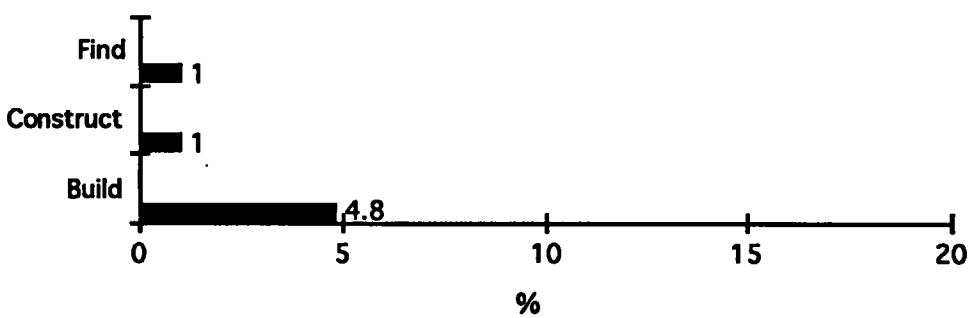

Chart 5.7 Rule inference engine rule base analysis.

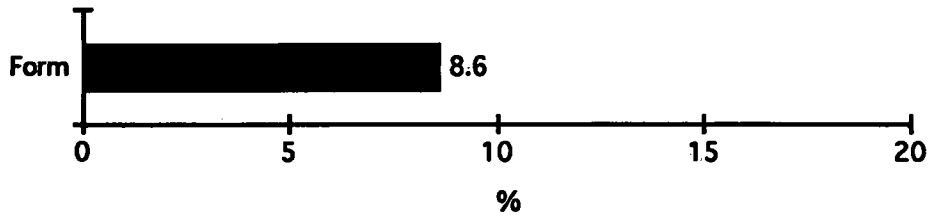

Chart 5.8 Rule inference engine analysis.

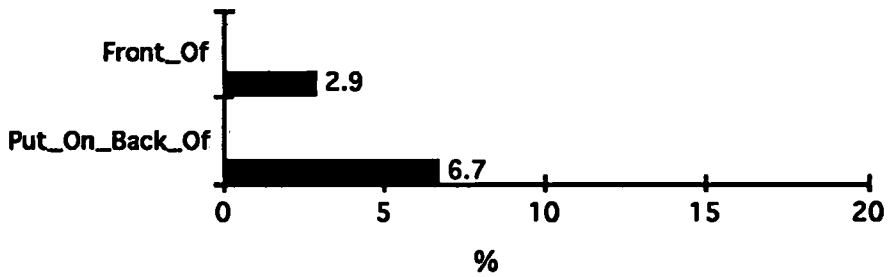

Chart 5.9 Rule inference engine list analysis.

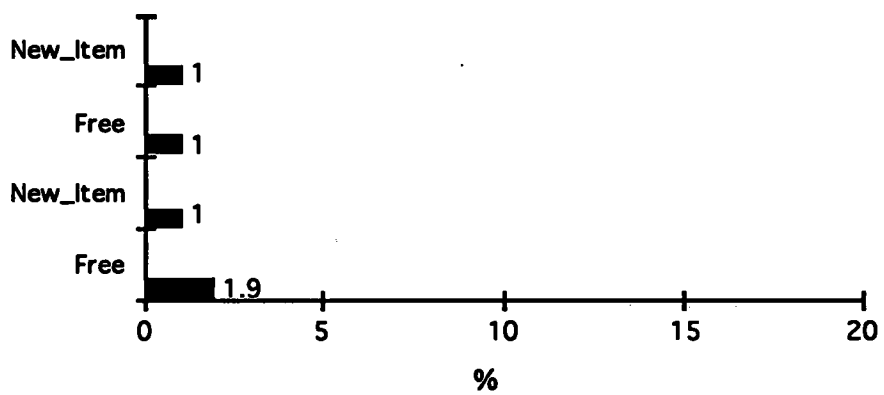

Chart 5.10 Rule inference engine free list analysis.

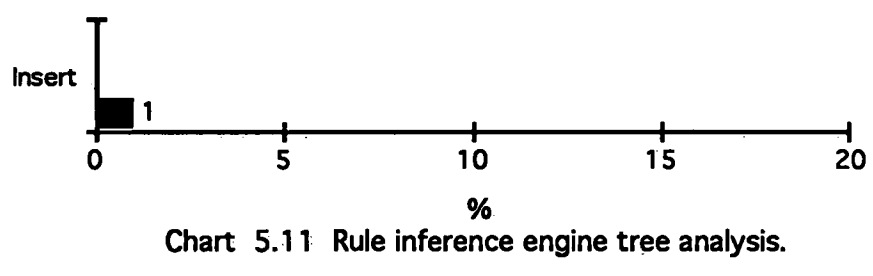

Chart 5.11 Rule inference engine tree analysis.

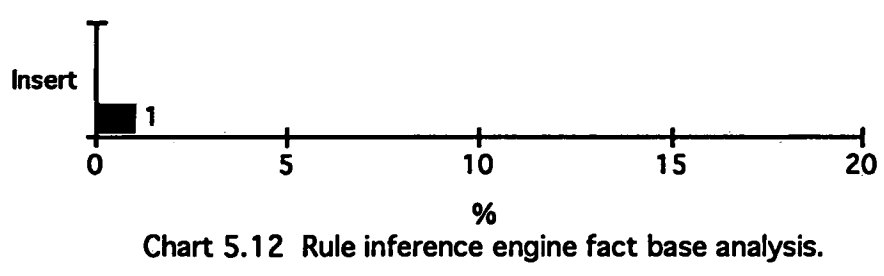

Chart 5.12 Rule inference engine fact base analysis.

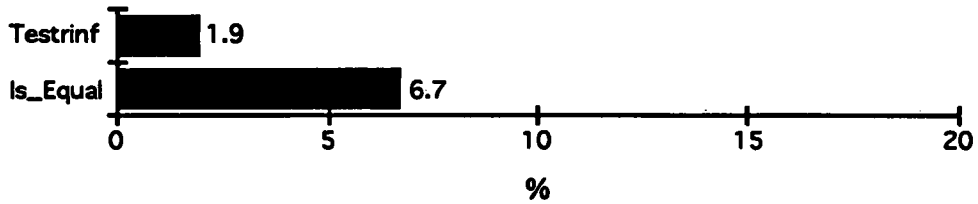

Chart 5.13 Testrinf analysis.

5.6 Summary

This chapter describes the implementation of a rule abstract knowledge type. A simple rule syntax is implemented, together with a means of forward chaining across a series of independent rule base instances. The implementation of the data structures and operations are much simpler than those needed to support the logic abstract knowledge type. A series of tests show that, like the logic abstraction, the majority of CPU time is spent In operations from Dynamic String; a total of 64.3% when building the rule base and 47.8% when querying the inference engine. Unlike the logic abstraction, which spends only 2.1% of time getting the clauses from the external file, a significant amount of time, 16.6%, is spent by the rule component on this task. The most significant rule base operation is Build where 5.0% of CPU time is spent; again small compared with the Dynamic String contribution.

Chapter 6

Implementation of a Frame Abstract Knowledge Type

6.1 Introduction

Chapter 2 identified frames as one of a range of structured knowledge representation paradigms. The representation is useful in cases where the application has an inherent hierarchical structure, and where there is a need to record the detailed characteristics which describe the problem domain. Furthermore, the representation is often used to integrate other knowledge representation paradigms such as rules. The aim of this chapter is to describe the implementation of a frame-based abstract knowledge type.

6.2 Frame Base Data Structures

The Frame Base data structure requires a set of nodes to represent an arbitrary number of frames. In addition, each frame can have an arbitrary number of slots, which in turn can have an arbitrary number of facets.

6.2.1 Frame Base Architecture

Frame Base is built from the nodes shown in Fig 6.1. The Facet node is generic to allow the user component to tailor the structure to meet application requirements.

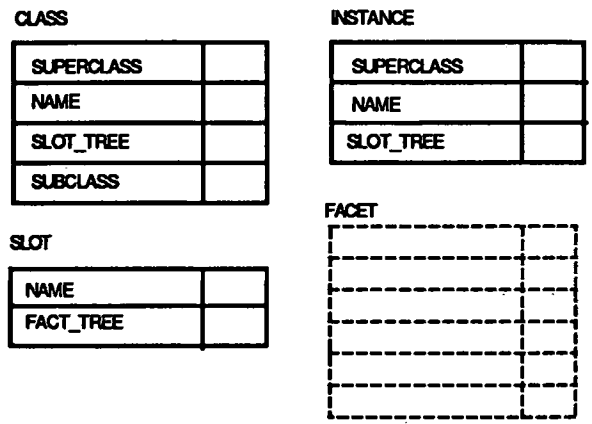

Fig 6.1 Frame base nodes.

6.2.2 Frame Tree and Frame Search Tree

Frame Base is Implemented as a tree of Class and Instance nodes connected as shown in Fig 6.2. Since the frames are kept in hierarchical order, and since each frame may have an arbitrary number of sub-frames, an N-ary branching tree is used to store the frames. However, an efficient frame search strategy is also required, consequently a balanced binary search tree is used to record the frame entries by overlaying the frame base structure.

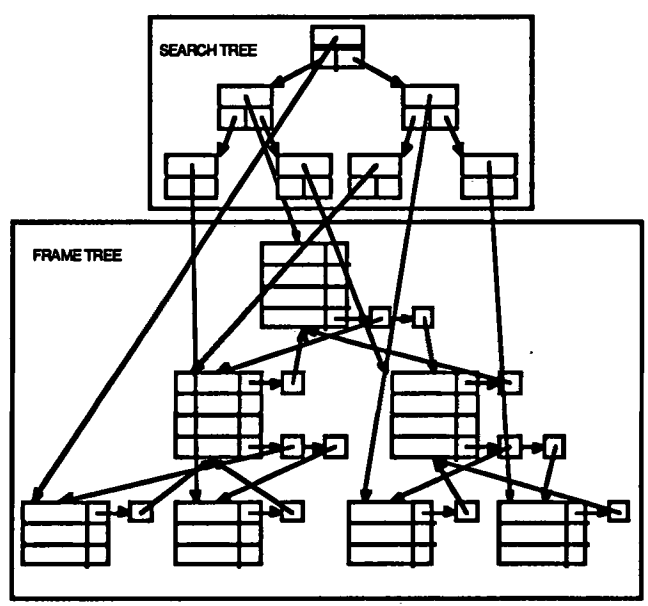

Fig 6.2 Frame and search trees.

6.2.3 Slot and Facet Trees

An example of the Slot and Facet Trees is shown in Fig 6.3. Since each slot and facet may be kept in any order, a balanced binary search tree is used for both the Slot and Facet trees.

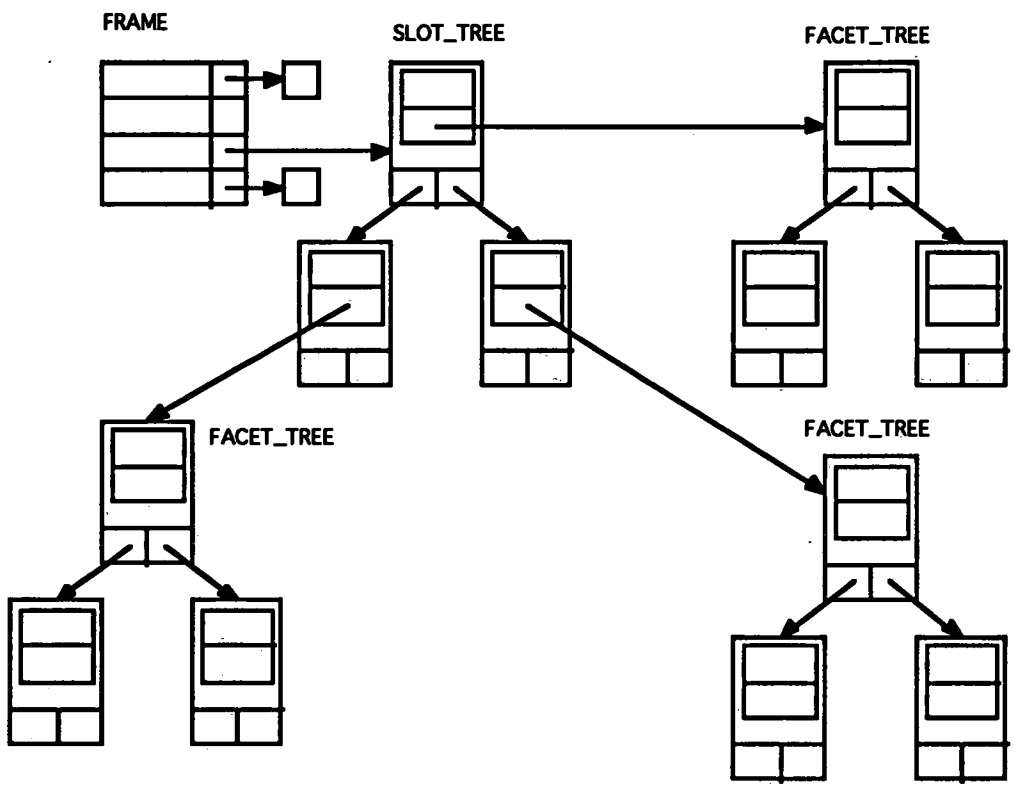

Fig 6.3 The slot and facet trees.

6.3 Frame Base Operations

A limited set of frame base operations are provided in order to build and traverse the structure. The Frame Base abstract knowledge type architecture is shown in Fig 6.4.

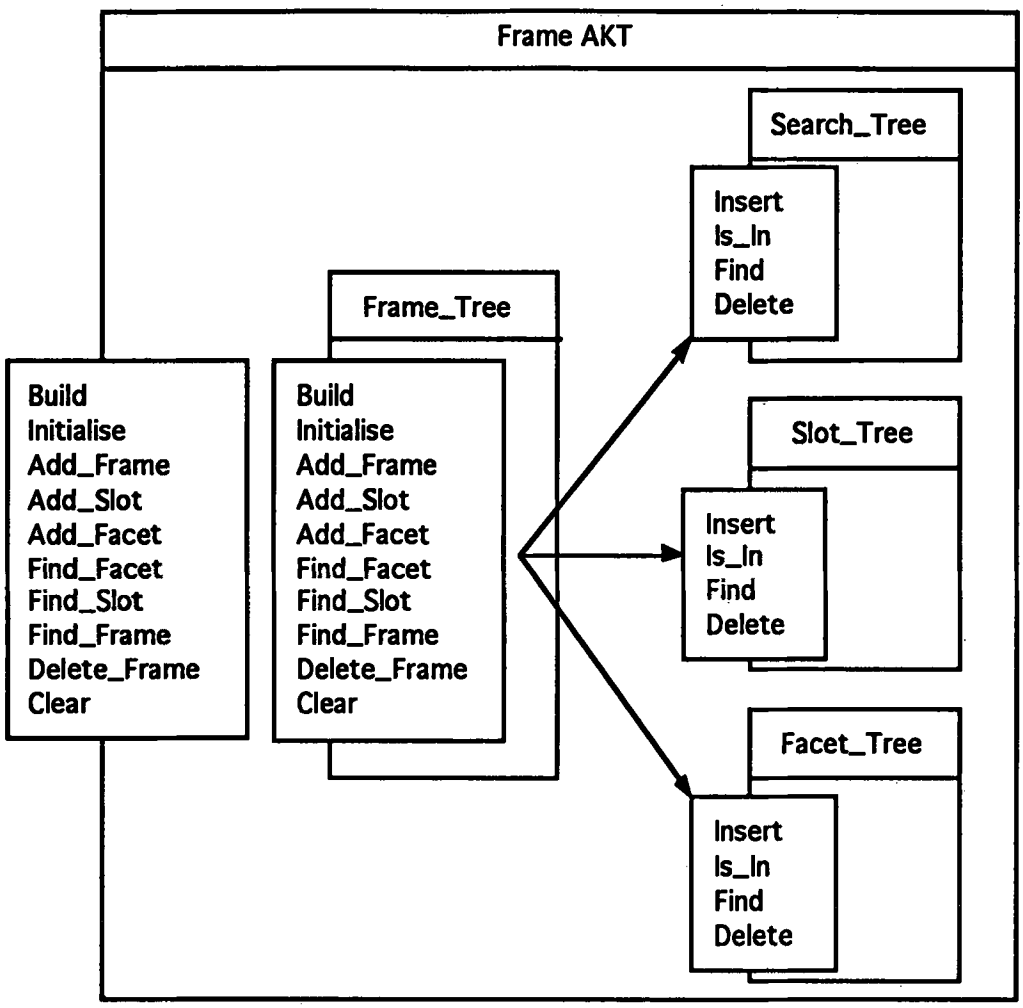

Fig 6.4 The frame abstract knowledge type architecture.

6.3.1 Build and Initialise

The Build and Initialise operations, unlike the other abstract knowledge types, where the build operations are called implicitly on instantiation, the Frame Base Build and Initialise operations are called explicitly by the application component; this is so that an empty instance may be created when attached to the blackboard. In this case, multiple frame base instances are instantiated as part of the blackboard architecture as well as stand-alone components. This strategy is described further in Chapter 8.

6.3.2 Frame Operations

A number of frame manipulation operations are provided and made visible to the application component. The operations enable addition and deletion of frames, slots and facets, together with the ability to search the frame base. An application component uses these primitive operations to develop application dependent frame base manipulation routines.

6.4 Analysis

6.4.1 Ada Package Structure

The Ada package dependency structure is shown in Fig 6.5; this shows further reuse of the generic packages Free List, Dynamic String, Tree, System Types and List. A single test harness, Testfkb, is provided to support the testing of the build process. The code listings are given in Annex A

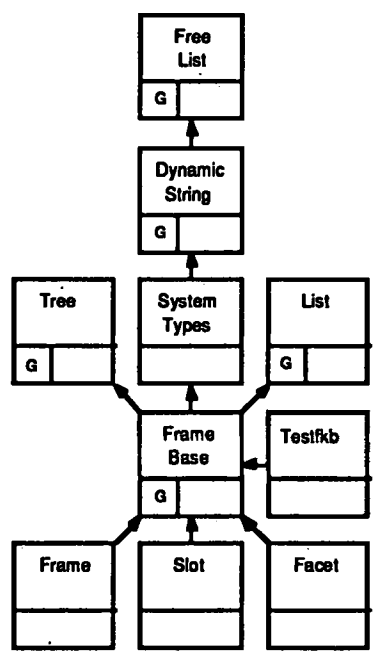

Fig 6.5 Frame abstract knowledge type Ada package dependencies.

A user component instantiates Frame Base to establish an instance of the frame abstract knowledge type. The generic specification of Frame Base is:

```
with   TEXT_IO,
       SYSTEM_TYPES_PACKAGE,
       GENERIC_TREE_PACKAGE,
       GENERIC_LIST_PACKAGE;
generic
       type SLOT_TYPE is (<>);
       type FACET_RECORD_TYPE(SLOT_KIND : SLOT_TYPE) is private;
       type FACET_RECORD_PTR_TYPE is access FACET_RECORD_TYPE;
       with function "<"(
       LEFT_FACET_PTR          : in FACET_RECORD_PTR_TYPE ;
       RIGHT_FACET_PTR         : in FACET_RECORD_PTR_TYPE) return BOOLEAN;
       with function ">"(
       LEFT_FACET_PTR          : in FACET_RECORD_PTR_TYPE ;
       RIGHT_FACET_PTR         : in FACET_RECORD_PTR_TYPE) return BOOLEAN;
       with function IS_EQUAL(
       FACET_NAME              : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
       FACET_PTR               : in FACET_RECORD_PTR_TYPE) return BOOLEAN;
       with function IS_LESS_THAN(
       FACET_NAME              : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
       FACET_PTR               : in FACET_RECORD_PTR_TYPE) return BOOLEAN;
       with function IS_GREATER_THAN(
       FACET_NAME              : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
       FACET_PTR               : in FACET_RECORD_PTR_TYPE) return BOOLEAN;
       with procedure PUT(
       FACET_PTR               : in FACET_RECORD_PTR_TYPE);
       with procedure GET(
       FACET_PTR               : in out FACET_RECORD_PTR_TYPE;
       FROM_FILE               : in out TEXT_IO.FILE_TYPE);
package GENERIC_FRAME_BASE_PACKAGE is ...
```

This permits instantiation of multiple, independent instances of the abstract knowledge type.

In order to make the abstract knowledge type flexible, the Facet structure is specified by the application component. The Facet generic formal parameters and associated generic operations are subsequently used to instantiate an instance of the generic Facet Tree.

6.4.2 Facet Specification

The specification of Facet uses the generic parameters above as actual generic parameters to instantiate Facet Tree.

```
package FACET_PACKAGE is
        package FACET_TREE_PACKAGE is new GENERIC_TREE_PACKAGE(
        ITEM_BASE_TYPE              =>  SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.
                                        STRING,
        ITEM_COMPOSITE_TYPE         =>  FACET_RECORD_TYPE,
        ITEM_PTR_TYPE               =>  FACET_RECORD_PTR_TYPE,
        "<"                         =>  "<",
        ">"                         =>  ">",
        IS_EQUAL                    =>  IS_EQUAL,
        IS_LESS_THAN                =>  IS_LESS_THAN,
        IS_GREATER_THAN             =>  IS_GREATER_THAN,
        PUT                         =>  PUT);
end FACET_PACKAGE;
```

6.4.3 Slot Specification

The specification of Slot requires the specification of the slot and slot operations prior to creating the Slot Tree instance.

```
package SLOT_PACKAGE is
    type SLOT_RECORD_TYPE is
    record
        NAME                :   SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.STRING;
        FACET_TREE_PTR      :   FACET_PACKAGE.
                                FACET_TREE_PACKAGE.
                                TREE_PTR_TYPE;
    end record;
    type SLOT_RECORD_PTR_TYPE is access SLOT_RECORD_TYPE;
    function IS_EQUAL(
    LEFT_SLOT_PTR       : in SLOT_RECORD_PTR_TYPE;
    RIGHT_SLOT_PTR      : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;
    function "<"(
    LEFT_SLOT_PTR       : in SLOT_RECORD_PTR_TYPE ;
    RIGHT_SLOT_PTR      : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;
    function ">"(
    LEFT_SLOT_PTR       : in SLOT_RECORD_PTR_TYPE ;
    RIGHT_SLOT_PTR      : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;
    function IS_EQUAL(
    SLOT_NAME       : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
```

```
        SLOT_PTR        : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;
        function IS_LESS_THAN(
        SLOT_NAME       : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        SLOT_PTR        : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;
        function IS_GREATER_THAN(
        SLOT_NAME       : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        SLOT_PTR        : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;

        procedure PUT(
        SLOT_PTR        : in SLOT_RECORD_PTR_TYPE);

        package SLOT_TREE_PACKAGE is new GENERIC_TREE_PACKAGE(
        ITEM_BASE_TYPE              =>  SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.
                                        STRING,
        ITEM_COMPOSITE_TYPE         =>  SLOT_RECORD_TYPE,
        ITEM_PTR_TYPE               =>  SLOT_RECORD_PTR_TYPE,
        "<"                         =>  "<",
        ">"                         =>  ">",
        IS_EQUAL                    =>  iS_EQUAL,
        IS_LESS_THAN                =>  IS_LESS_THAN,
        IS_GREATER_THAN             =>  IS_GREATER_THAN,
        PUT                         =>  PUT);
    end SLOT_PACKAGE;
```

6.4.4 Frame Specification

The specification of Frame requires the specification of the frame and frame operations prior to creating the Frame Tree instance.

```
package FRAME_PACKAGE is

    type FRAME_KIND is (CLASS, INSTANCE);
    type FRAME_RECORD_TYPE(KIND : FRAME_KIND);
    type FRAME_RECORD_PTR_TYPE is access FRAME_RECORD_TYPE;

    function IS_EQUAL(
    LEFT_FRAME_PTR      : in FRAME_RECORD_PTR_TYPE;
    RIGHT_FRAME_PTR     : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

    package LIST_PACKAGE is new GENERIC_LIST_PACKAGE(
    ITEM_TYPE  => FRAME_RECORD_PTR_TYPE,
    IS_EQUAL   => IS_EQUAL);

    function "<"(
    LEFT_FRAME_PTR      : in FRAME_RECORD_PTR_TYPE ;
    RIGHT_FRAME_PTR     : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;
    function ">"(
    LEFT_FRAME_PTR      : in FRAME_RECORD_PTR_TYPE ;
```

```
        RIGHT_FRAME_PTR        : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;
        function IS_EQUAL(
        FRAME_NAME      : In SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FRAME_PTR       : In FRAME_RECORD_PTR_TYPE) return BOOLEAN;
        function IS_LESS_THAN(
        FRAME_NAME      : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FRAME_PTR       : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;
        function IS_GREATER_THAN(
        FRAME_NAME      : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FRAME_PTR       : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;
        procedure PUT(
        FRAME_PTR       : in FRAME_RECORD_PTR_TYPE);

        type FRAME_RECORD_TYPE(KIND : FRAME_KIND) is
        record
              SUPERCLASS_LIST       :     LIST_PACKAGE.LIST_TYPE;
              NAME                  :     SYSTEM_TYPES_PACKAGE.
                                          DYNAMIC_STRING.STRING;
              SLOT_TREE_PTR         :     SLOT_PACKAGE.
                                          SLOT_TREE_PACKAGE.
                                          TREE_PTR_TYPE;
              case KIND is
                    when CLASS      => SUBCLASS_LIST  : LIST_PACKAGE.LIST_TYPE;
                    when INSTANCE   => null;
              end case;
        end record;

        package FRAME_TREE_PACKAGE is new GENERIC_TREE_PACKAGE(
        ITEM_BASE_TYPE              =>    SYSTEM_TYPES_PACKAGE.
                                          DYNAMIC_STRING.STRING,
        ITEM_COMPOSITE_TYPE         =>    FRAME_RECORD_TYPE,
        ITEM_PTR_TYPE               =>    FRAME_RECORD_PTR_TYPE,
        "<"                         =>    "<",
        ">"                         =>    ">",
        IS_EQUAL                    =>    IS_EQUAL,
        IS_LESS_THAN                =>    IS_LESS_THAN,
        IS_GREATER_THAN             =>    IS_GREATER_THAN,
        PUT                         =>    PUT);
end FRAME_PACKAGE;
```

6.4.5 Frame Base Specification

Having established Facet, Slot and Frame, Frame Base is specified as:

```
        type FRAME_BASE_RECORD_TYPE is private;

        type FRAME_BASE_RECORD_TYPE is
        record
              FRAME_BASE_PTR        :     FRAME_PACKAGE.
                                          FRAME_RECORD_PTR_TYPE;
```

```
        FRAME_SEARCH_TREE   :    FRAME_PACKAGE.
                                 FRAME_TREE_PACKAGE.
                                 TREE_PTR_TYPE;
    end record;
```

6.4.6 Specification of Frame Base Operations

Frame Base requires a number of low level operations which are specified as:

```
    procedure BUILD(
    FRAME_BASE_RECORD          : in out    FRAME_BASE_RECORD_TYPE;
    FRAME_FILE_LIST_FILENAME   : in        STANDARD.
                                           STRING);
    procedure INITIALISE_FRAME_BASE(
    FRAME_BASE_RECORD          : in out    FRAME_BASE_RECORD_TYPE);
    procedure ADD_FRAME(
    FRAME_NAME          : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_TYPE          : in        FRAME_PACKAGE.
                                    FRAME_KIND :=
                                    FRAME_PACKAGE.CLASS;
    SUPERCLASS_NAME     : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_BASE_RECORD   : in out    FRAME_BASE_RECORD_TYPE);
    procedure ADD_SLOT(
    SLOT_NAME           : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_NAME          : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_BASE_RECORD   : in out    FRAME_BASE_RECORD_TYPE);
    procedure ADD_FACET(
    FACET_PTR           : in        FACET_RECORD_PTR_TYPE;
    SLOT_NAME           : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_NAME          : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_BASE_RECORD   : in out    FRAME_BASE_RECORD_TYPE);
    procedure FIND_FACET(
    FACET_NAME          : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    SLOT_NAME           : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_NAME          : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_BASE_RECORD   : in        FRAME_BASE_RECORD_TYPE;
    FACET_PTR           : in out    FACET_RECORD_PTR_TYPE);
    procedure FIND_SLOT(
    SLOT_NAME           : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
    FRAME_NAME          : in        SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.STRING;
```

```
FRAME_BASE_RECORD   : in       FRAME_BASE_RECORD_TYPE;
SLOT_PTR            : in out   SLOT_RECORD_PTR_TYPE);
procedure FIND_FRAME(
FRAME_NAME          : in       SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.STRING;
FRAME_BASE_RECORD   : in       FRAME_BASE_RECORD_TYPE;
FRAME_PTR           : in out   FRAME_RECORD_PTR_TYPE);
procedure ADD_FRAME(
FRAME_PTR           : in       FRAME_PACKAGE.
                               FRAME_RECORD_PTR_TYPE;
SUPERCLASS_NAME     : in       SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.STRING;
FRAME_BASE_RECORD   : in out   FRAME_BASE_RECORD_TYPE);
procedure DELETE_FRAME(
FRAME_PTR           : in       FRAME_PACKAGE.
                               FRAME_RECORD_PTR_TYPE;
SUPERCLASS_NAME     : in       SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.STRING;
FRAME_BASE_RECORD   : in out   FRAME_BASE_RECORD_TYPE);
procedure CLEAR(
FRAME_BASE_RECORD   : in out   FRAME_BASE_RECORD_TYPE);
```

6.5 Results

Test results for a Frame Base containing 4650 frames are recorded in Table 6.1 and Charts 6.1 to 6.7, but discussion of the results is deferred to Chapter 9.

Item	Lines of Code	CPU Time %	Ada new
Free List	49	0	0
Dynamic String	828	17.7	49200
List	520	1.3	16500
Tree	526	4.3	108312
System Types	397	0	0
Frame Base	1340	69.4	40950
Frame	166	0.9	0
Slot	151	0.1	0
Facet	15	0	0
Test Types	84	4.2	0
Min:Sec		0:38.85	

Table 6.1 Frame base test results.

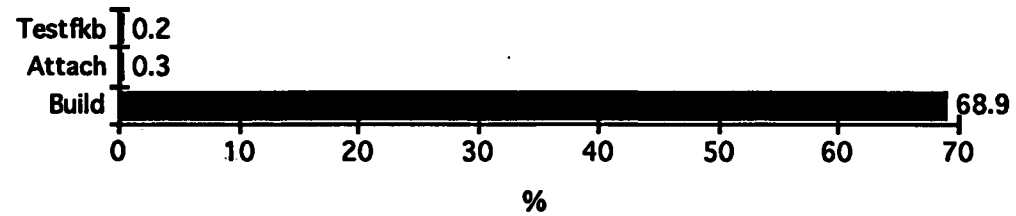

Chart 6.1 Frame base analysis.

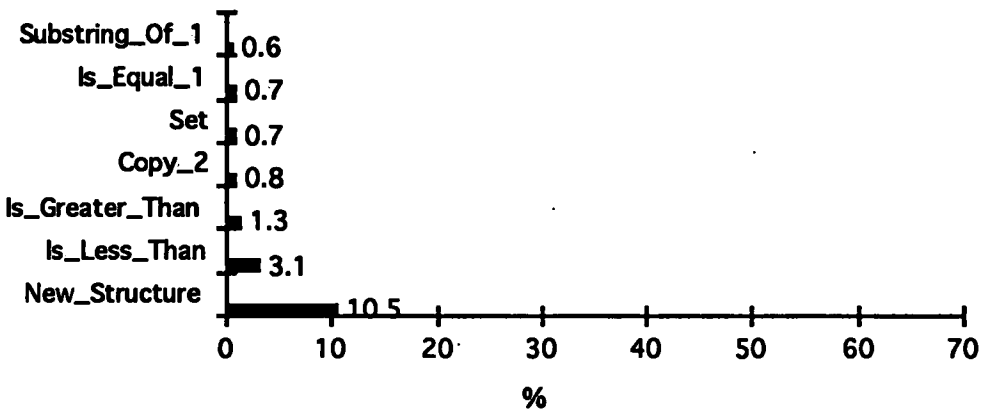

Chart 6.2 Frame base dynamic string analysis.

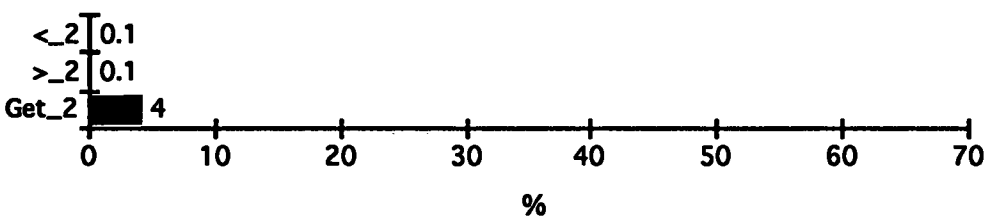

Chart 6.3 Frame base test types analysis.

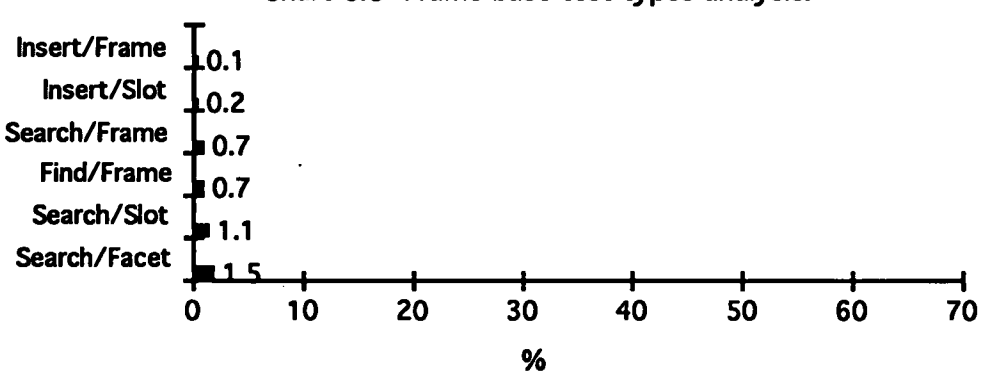

Chart 6.4 Frame base tree analysis.

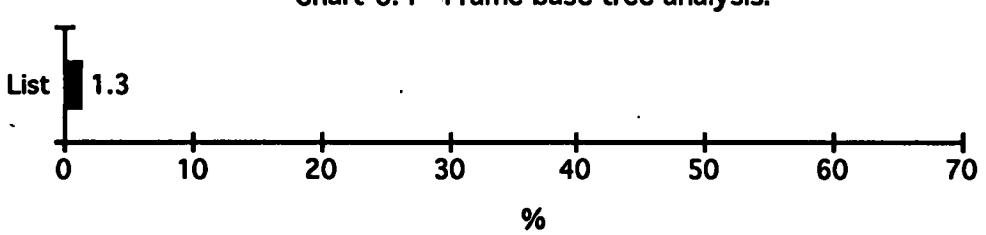

Chart 6.5 Frame base list analysis.

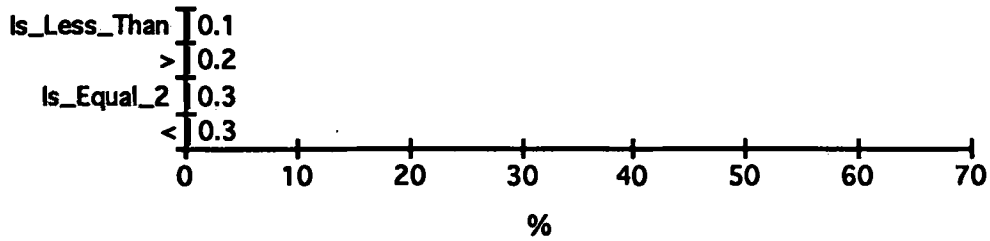

Chart 6.6 Frame base frame analysis.

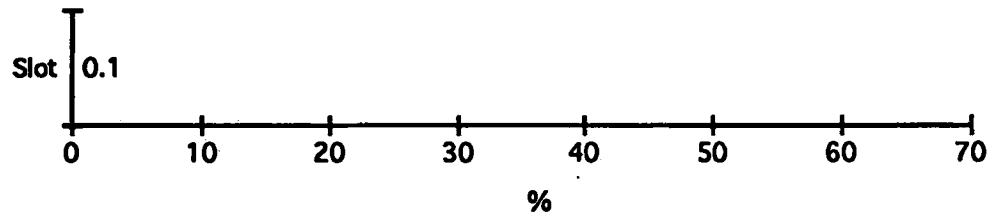

Chart 6.7 Frame base slot analysis.

6.6 Summary

This chapter describes the implementation of a frame abstract knowledge type. In particular, it describes the data structures that are used to implement the abstraction. Due to the generic structure of the integrated components, the instantiation is complex and involves the use of nested subprogram instances. Unlike the logic and rule abstract knowledge types, where Dynamic String dominates the processes, the frame component test shows that the frame Build operation consumes 68.9% of the CPU time, with a single dynamic allocation statement taking 45.6%.

Chapter 7

Implementation of a Blackboard Abstract Knowledge Type

7.1 Introduction

Chapter 3 concluded that a blackboard architecture is a suitable framework on which to integrated multiple and diverse knowledge representation paradigms. Furthermore, Chapter 4 described the blackboard framework in some detail and showed that a blackboard system consists of three main components: first, the blackboard, a framework on which the blackboard entries are placed; second, knowledge sources, which encapsulate domain knowledge and are responsible for changing the blackboard entries; third, a control mechanism which schedules knowledge source access to the blackboard. The aim of this chapter is to describe how a generic blackboard abstract knowledge type was implemented.

7.2 Blackboard Data Structures

A generally applicable blackboard abstract knowledge type needs to be an unconstrained generic space on to which blackboard entries may be placed. In particular, the number of levels, the number of horizontal divisions within a level, and the structure of each level entry need to be generic and unconstrained so that user components may tailor the blackboard to meet precise application requirements.

The top-level view of such an abstract blackboard is represented by the matrix shown in Fig 7.1.

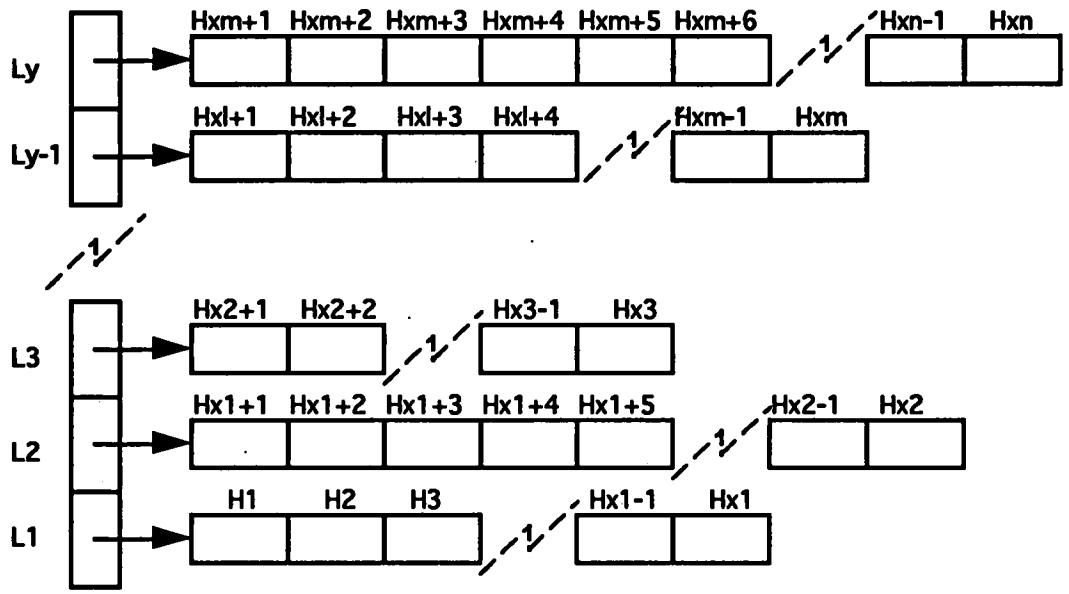

Fig 7.1 An unconstrained generic blackboard space.

In general terms, the vertical and horizontal dimensions of this arbitrary matrix are defined by the generic parameters:

```
type Generic_Level_Index_Type is (<>);
type Generic_Horizontal_Index_Type is (<>);
```

This definition permits the range of each vertical and horizontal dimension to be different and specified by an application component. The formal generic parameters specified by (<>) indicate that the actual generic parameters may be any discrete type; a discrete type is necessary since the values of the discrete type are used to set the vertical and horizontal array dimensions. The actual generic parameters for Generic_Level_Index_Type and Generic_Horizontal_Index_Type are application defined enumeration type names, which specify the range of values for each array.

For example,

```
type Application_Level_Index_Type is (L1, L2, L3, ..., Ly-1, Ly);

type Application_Horizontal_Index_Type is (
        Hxm+1, Hxm+2, Hxm+3, Hxm+4, Hxm+5, ..., Hxn-1, Hxn,
        ---, ---,
        Hx1+1, Hx1+2, Hx1+2, Hx1+3, Hx1+4, Hx1+5, ..Hx2-1, Hx2,
        H1, H2, H3, ..., Hx1-1, Hx1);
```

where L and H are application identifiers that define the vertical and horizontal index values. An initialisation algorithm, provided by the application component, is responsible for creating the detailed framework from such a specification.

Two further generic formal parameters are needed to complete the generic definition of the blackboard; first, the specification of the generic abstract structures which are to be placed in each blackboard location; second, an access type to these structures. For example,

```
type Generic_Abstract_Structure_Type is private;
type Generic_Abstract_Structure_Pointer_Type is access
        Generic_Abstract_Structure_Type;
```

The private specification in this definition permits the matching actual generic parameter to be any Ada type that supports the operations of assignment and equality; an access type is needed since the abstract structures are created dynamically as the matrix is initialised to fit the application requirement.

Consequently, the complete abstract blackboard generic specification is:

```
generic
        type Generic_Abstract_Structure_Type is private;
        type Generic_Abstract_Structure_Pointer_Type is access
                Generic_Abstract_Structure_Type;
        type Generic_Level_Index_Type is (<>);
        type Generic_Horizontal_Index_Type is (<>);
package Generic_Blackboard is ...
```

The application abstract structures are defined using a discriminant specification:

```
type Application_Abstract_Structure_Kind is
      (ADT_1, ADT_2, ..., ADT_n, AKT_1, AKT_2, ..., AKT_n);

type Application_Abstract_Structure_Type
      (Kind : Application_Abstract_Structure_Kind) is
record
      case Kind is
            when ADT_1      => ADT1    : ADT_Type_1;
            when ADT_2      => ADT2    : ADT_Type_2;
            ...
            when ADT_n      => ADTn    : ADT_Type_n;
            when AKT_1      => AKT1    : AKT_Type_1;
            when AKT_2      => AKT2    : AKT_Type_2;
            ...
            when AKT_N      => AKTn    : AKT_Type_n;
      end case;
end record;

type  Application_Abstract_Structure_Pointer_Type is access
      Application_Abstract_Structure_Type;
```

From this specification, it can be seen that the user component defined blackboard structures are any ADT or AKT available to the application. It is worth noting that the use of abstract knowledge types in this role was not anticipated at the start of the research.

An instance of the generic blackboard is instantiated with the statement:

```
package Application_Blackboard is new Generic_Blackboard(
      Generic_Abstract_Structure_Type                    =>
            Application_Abstract_Structure_Type
      Generic_Abstract_Structure_Pointer_Type            =>
            Application_Abstract_Structure_Pointer_Type,
      Generic_Level_Index_Type                           =>
            Application_Level_Index_Type,
      Generic_Horizontal_Index_Type                      =>
            Application_Horizontal_Index_Type);
```

At the point of instantiation, the generic formal parameters are replaced by the user component actual parameters to form a blackboard tailored to meet the

application requirement. For example, a blackboard level instantiated with list ADTs is shown in Fig 7.2.

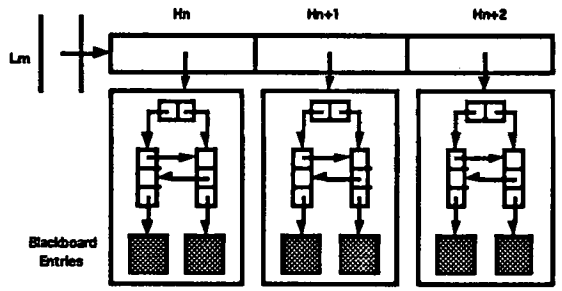

Fig 7.2 A blackboard level.

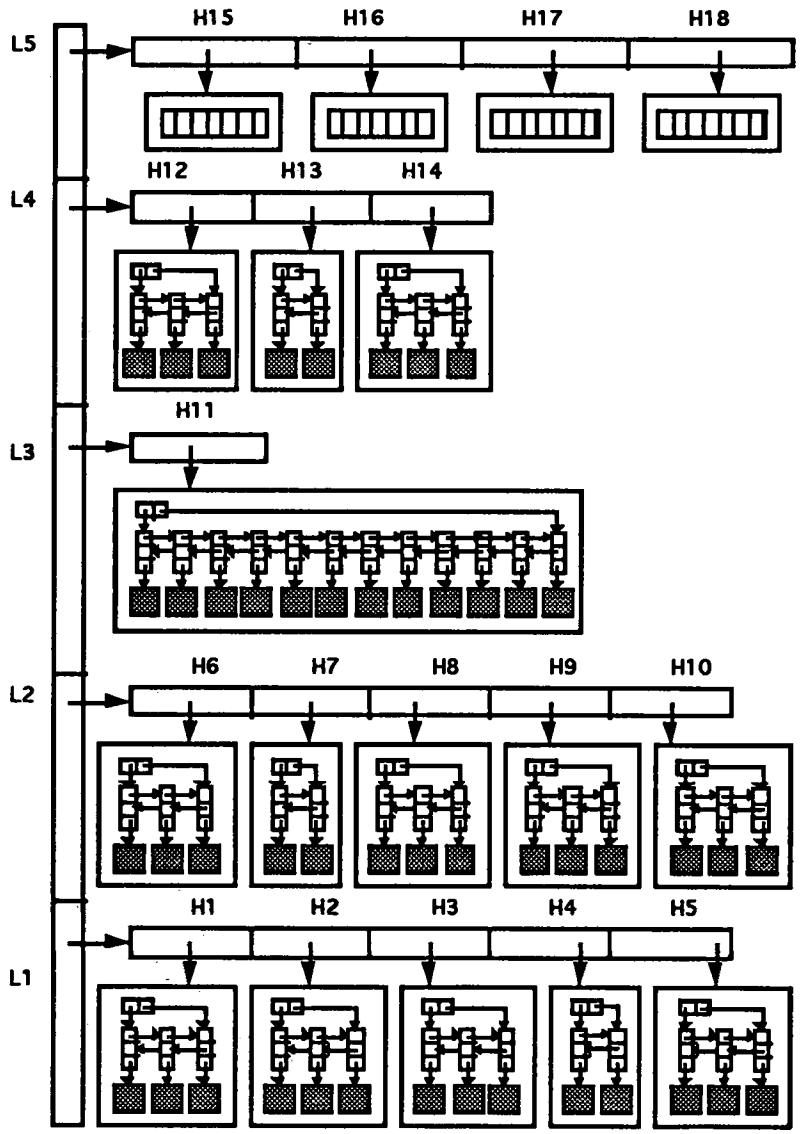

Fig 7.3 An abstract blackboard.

An example of an abstract blackboard instance, with list and array actual generic structures, is shown in Fig 7.3; some detail has been omitted in order to simplify the presentation. In addition, the links between levels are not shown; these are determined by the application. An example of a practical blackboard is given in Chapter 8.

7.3 Blackboard Operations

The generic operations to implement the blackboard structure are limited in number, since blackboards are application dependent. The simple blackboard abstract knowledge type is shown in Fig 7.4.

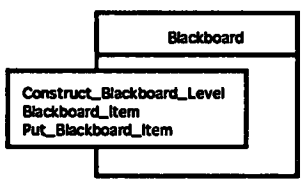

Fig 7.4 The blackboard abstract knowledge type.

Construct_Blackboard_Level allocates memory to a single blackboard level, constraining the upper and lower bounds of the level to values specified by the user component. Blackboard_Item and Put_Blackboard_Item retrieve and assign items to the blackboard level locations. The code listings are given in Annex A.

7.4 Analysis

7.4.1 Ada Package Structure

The Ada package dependencies are shown in Fig 7.5.

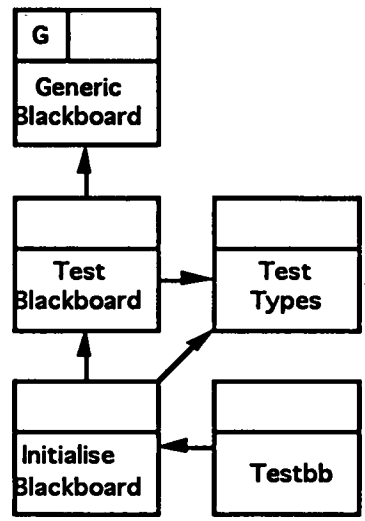

Fig 7.5 Blackboard abstract knowledge type Ada package dependencies.

As can be seen from the figure, four components are provided by the user, in this case the test harness. Test Types provide the application dependent definitions from which the blackboard is constructed. Test Blackboard is the instantiation of the generic component using the actual generic parameters defined in Test Types. Initialise Blackboard encapsulates the user defined knowledge which constructs the blackboard levels to meet the user component requirement; in practice, this operation is implemented as a knowledge source. Testbb is a simple test driver.

7.4.2 Blackboard Specification

A user component instantiates Generic Blackboard to establish an instance of the blackboard abstract knowledge type. As indicated earlier, the generic specification for Generic Blackboard requires four parameters. These are implemented as:

```
generic
        type ITEM_STRUCTURE_TYPE is private;
        type ITEM_STRUCTURE_PTR_TYPE is access ITEM_STRUCTURE_TYPE;
        type LEVEL_INDEX_TYPE is (<>);
        type HORIZONTAL_INDEX_TYPE is (<>);
```

```
package GENERIC_BLACKBOARD_PACKAGE is

    type BLACKBOARD_HORIZONTAL_ARRAY_TYPE is array
    (HORIZONTAL_INDEX_TYPE range <>) of ITEM_STRUCTURE_PTR_TYPE;

    type BLACKBOARD_HORIZONTAL_PTR_TYPE is access
    BLACKBOARD_HORIZONTAL_ARRAY_TYPE;

    type BLACKBOARD_TYPE is array
    (LEVEL_INDEX_TYPE'FIRST .. LEVEL_INDEX_TYPE'LAST) of
    BLACKBOARD_HORIZONTAL_PTR_TYPE;
```

In addition, the specification shows the definitions for the vertical and horizontal divisions of the blackboard; the vertical array is constrained by Level_Index_Type and the horizontal arrays by sections of Horizontal_Index_Type.

7.4.3 Specification of Blackboard Operations

The three blackboard operations are specified as:

```
procedure CONSTRUCT_BLACKBOARD_LEVEL(
BLACKBOARD      : in out    BLACKBOARD_TYPE;
LEVEL           : in        LEVEL_INDEX_TYPE;
FROM            : in        HORIZONTAL_INDEX_TYPE;
TO              : in        HORIZONTAL_INDEX_TYPE);

function BLACKBOARD_ITEM(
BLACKBOARD          : in    BLACKBOARD_TYPE;
LEVEL_INDEX         : in    LEVEL_INDEX_TYPE;
HORIZONTAL_INDEX    : in    HORIZONTAL_INDEX_TYPE)
return ITEM_STRUCTURE_PTR_TYPE;

procedure PUT_BLACKBOARD_ITEM(
BLACKBOARD          : in    BLACKBOARD_TYPE;
ITEM_STRUCTURE      : in    ITEM_STRUCTURE_PTR_TYPE;
LEVEL_INDEX         : in    LEVEL_INDEX_TYPE;
HORIZONTAL_INDEX    : in    HORIZONTAL_INDEX_TYPE);
```

7.4.4 Test Types Specification

A set of abstract test types is provided for component testing. These are:

```
with  SYSTEM_TYPES_PACKAGE,
      GENERIC_LIST_PACKAGE;
package TEST_BB_TYPES_PACKAGE is

      -- Vertical blackboard divisions
      type TEST_BB_LEVEL_INDEX_TYPE Is (
            L5,
            L4,
            L3,
            L2,
            L1);

      -- Horizontal blackboard divisions
      type TEST_BB_HORIZONTAL_INDEX_TYPE is (
            H15, H16, H17, H18,
            H12, H13, H14,
            H11,
            H6, H7, H8, H9, H10,
            H1, H2, H3, H4, H5);


      -- Blackboard entry types
      type TEST_BB_NODE_KIND_TYPE is (
            BB_NODE_TYPE_1,
            BB_NODE_TYPE_2,
            BB_NODE_TYPE_3);

      type TEST_BB_NODE_RECORD(
      KIND : TEST_BB_NODE_KIND_TYPE := BB_NODE_TYPE_1);
      type TEST_BB_NODE_PTR_TYPE is access TEST_BB_NODE_RECORD;
      function IS_EQUAL(
      TEST_BB_NODE_PTR_1 : in TEST_BB_NODE_PTR_TYPE;
      TEST_BB_NODE_PTR_2 : in TEST_BB_NODE_PTR_TYPE) return BOOLEAN;

      package TEST_BB_LIST_PACKAGE
      is new GENERIC_LIST_PACKAGE(TEST_BB_NODE_PTR_TYPE, IS_EQUAL);

      type TEST_BB_ARRAY_TYPE is array(1..7) of TEST_BB_NODE_PTR_TYPE;

      type TEST_BB_NODE_RECORD
      (KIND : TEST_BB_NODE_KIND_TYPE := BB_NODE_TYPE_1) is
      record
            SUPPORTS          : TEST_BB_LIST_PACKAGE.LIST_TYPE;
            SUPPORTERS        : TEST_BB_LIST_PACKAGE.LIST_TYPE;
            case KIND is
                  when BB_NODE_TYPE_1 => null; -- definition of
                  when BB_NODE_TYPE_2 => null; -- application dependent
                  when BB_NODE_TYPE_3 => null; --. blackboard entries
            end case;
      end record;

      type TEST_BB_ITEM_STRUCTURE_TYPE is (BB_LIST, BB_ARRAY);
      type TEST_BB_ITEM_STRUCTURE_RECORD
      (ITEM_STRUCTURE_KIND : TEST_BB_ITEM_STRUCTURE_TYPE := BB_LIST) is
      record
```

```
        case ITEM_STRUCTURE_KIND is
        when BB_LIST     => BB_LIST      :    TEST_BB_LIST_PACKAGE.
                                              LIST_TYPE;
        when BB_ARRAY    => BB_ARRAY     :    TEST_BB_ARRAY_TYPE :=
                                              (1..7 => new
                                              TEST_BB_NODE_RECORD
                                              (BB_NODE_TYPE_1));
        end case;
    end record;

    type TEST_BB_ITEM_STRUCTURE_PTR_TYPE is access
        BLACKBOARD_ITEM_STRUCTURE_RECORD;
```

7.5 Test Blackboard Specification

An instance of Generic Blackboard is instantiated in Test Blackboard as follows:

```
with TEST_BB_TYPES_PACKAGE,
     GENERIC_BLACKBOARD_PACKAGE;
package TEST_BB_PACKAGE is
     package TEST_BLACKBOARD is new GENERIC_BLACKBOARD_PACKAGE
     (ITEM_STRUCTURE_TYPE       =>  TEST_BB_TYPES_PACKAGE.
                                    TEST_BB_ITEM_STRUCTURE_RECORD,
     ITEM_STRUCTURE_PTR_TYPE    =>  TEST_BB_TYPES_PACKAGE.
                                    TEST_BB_ITEM_STRUCTURE_PTR_TYPE,
     LEVEL_INDEX_TYPE           =>  TEST_BB_TYPES_PACKAGE.
                                    TEST_BB_LEVEL_INDEX_TYPE,
     HORIZONTAL_INDEX_TYPE      =>  TEST_BB_TYPES_PACKAGE.
                                    TEST_BB_HORIZONTAL_INDEX_TYPE);
```

7.6 Test Blackboard Initialisation Specification

The application test strategy to implement the framework shown in Fig 7.3 is specified as:

```
with TEST_BB_TYPES_PACKAGE,
     TEST_BB_PACKAGE;
package TEST_BB_INITIALISE_KS_PACKAGE is
     procedure BUILD_BLACKBOARD_LEVELS(
          BLACKBOARD : in out    TEST_BB_PACKAGE.
                                 TEST_BLACKBOARD.
                                 BLACKBOARD_TYPE);
```

7.7 Results

The following statistics represent the results of repeatedly building the specified blackboard 1000 times. The results are discussed in Chapter 9.

Item	Lines of Code	CPU Time %	Ada new
Generic Blackboard	140		
Test Blackboard	28	20.4	1400
Test Types	122		
Initialise Blackboard	175	26.8	1800
Testbb	17	0.8	
Min:Sec		0:06	

Table 7.1 Blackboard test results.

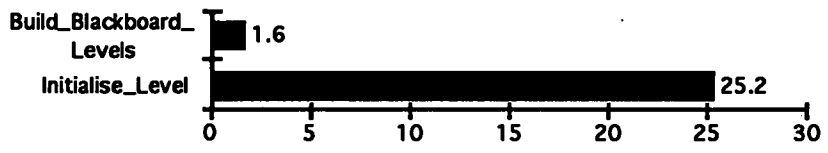

Chart 7.1 Initialise Blackboard.

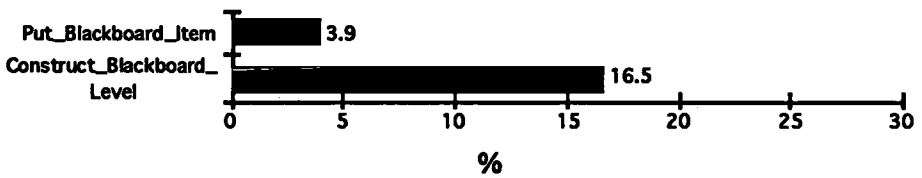

Chart 7.2 Test Blackboard.

7.8 Summary

This chapter describes the structure and implementation of a blackboard framework. In particular, it shows how a generic blackboard space, specified by unconstrained arrays, is constrained to meet a particular application requirement; in this case a test harness. The generic blackboard operations are simple constructors, since the application knowledge required to mould the framework to a particular shape is provided by the application component. In practice, this information is provided by a knowledge source. As expected, the analysis shows that the level allocators consume the majority of CPU time.

Chapter 8

Integrating Abstract Knowledge Types

8.1 Introduction

Chapters 4 to 6 described how three diverse generic abstract knowledge types, capable of multiple and independent instantiation, can be implemented. In addition, Chapter 7 described the implementation of a generic blackboard framework which can be tailored to represent specific application solution space requirements. The aim of this chapter is to show how multiple instances of the independent abstract knowledge types can be integrated using the generic blackboard framework, together with its associated knowledge sources, to solve a practical problem; a university timetable production domain is used to illustrate the strategy.

8.2 Integration Strategies

At the start of the research it was anticipated that the knowledge representation paradigms would be encapsulated as abstract knowledge types and instantiated as components in the blackboard knowledge sources. However, on completing the first abstract knowledge type it became clear that the components can be instantiated at any point in the blackboard architecture. In particular, the abstract knowledge types can be single or multiple components of the knowledge sources, independent components serving one or more knowledge sources, or components placed on the blackboard. These strategies are illustrated in Fig 8.1.

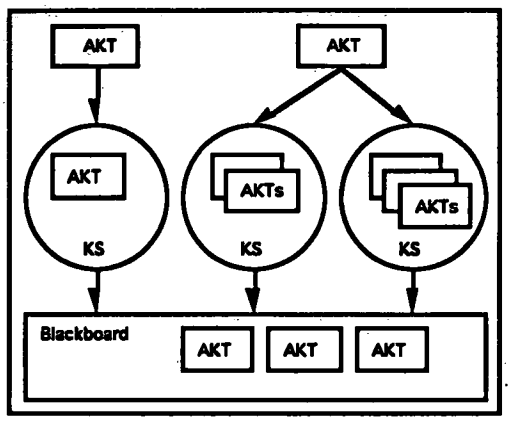

Fig 8.1 Abstract knowledge type integration strategies.

8.3 Integration Experiment

In order to test the integration strategies, a problem was needed that had sufficient representational complexity to warrant the application of multiple, diverse and communicating abstract knowledge types; a knowledge-rich problem domain was required. Since much work had been done in the past on a data fusion model [106], it was hoped that the model would provide a suitable test vehicle; unfortunately, the model has limited scope for the application of diverse knowledge representation paradigms. However, being closely involved with the manual production of university timetables, the timetable production problem domain is known to be a knowledge-rich environment. Consequently, the production of a prototype automated university timetable system is used as the integration test domain.

8.4 Domain Knowledge

The problem domain is the existing manual timetable production process as carried out at the Royal Military College of Science, Shrivenham. The College presents several undergraduate engineering degrees, which share many common modules throughout a three year programme. In order to reduce the problem to a

manageable size, it was decided that knowledge would be collected from only two of the current degrees covering one terms timetable; the Information Technology(IT) and Electronic Systems Engineering(ESE) degrees were chosen. This approach is considered valid, since each of the three terms have similar timetable structures and other degree profiles are similar to the two degrees that have been selected. Consequently, extending the scope of the prototype is a matter of knowledge acquisition rather than of additional conceptual complexity.

The first year of each degree comprise the academic modules shown in Table 8.1 and 8.2.

Module Name	Code
Mathematics I	E101
Signal Processing and Control	E151
Electrotechnology	E152
Electronics and Telecommunications I	E153
Computer Studies	E103
Stress Analysis and Engineering Materials	E102
Fluid Mechanics and Thermodynamics	E131
Engineering Mechanics	E132
Engineering Drawing, Design and Manufacture	E133
Engineer in Society I and II	E170

Table 8.1 ESE Part 1 modules.

Module Name	Code
Mathematics	I111
Resource Management	I113
Computer Programming	I114
Data Characteristics and Structures	I115
Computer Systems Principles and Architectures	I116
Electronics and Communications	I117
Discrete Mathematics	I118
Signal Processing and Control	I119
Data Analysis and Applied Probability	I120
Fundamentals of Systems	I122
Software Systems Design	I123
Human Implications of IT	I125

Table 8.2 IT Part 1 modules.

Each module is managed by a module manager. A module manager is responsible for determining the structure of the module, together with the lecture, tutorial, practical and staffing requirement for the module, and communicating this requirement to the member of staff responsible for co-ordinating timetable production. The timetable co-ordinator is responsible for identifying the commonality between degree codes and subsequently allocating timetable periods to meet the requirement requested by the module managers. A separate timetable is prepared for each degree showing the module codes appropriate to the degree, together with codes of common modules. The timetable production process is shown in Fig 8.2.

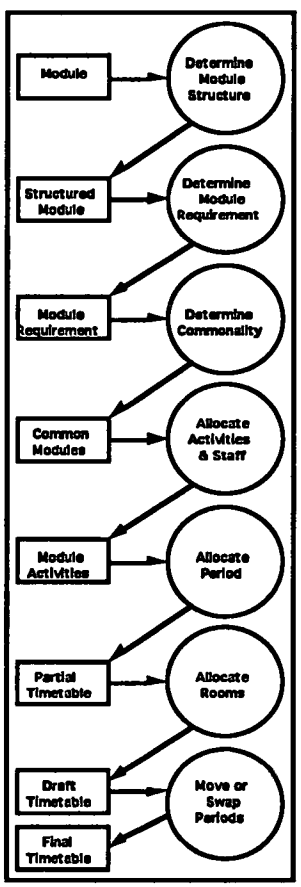

Fig 8.2 Timetable production process.

On completion of the timetable period allocation process, the draft timetables are passed by the timetable co-ordinator to the academic registry, where rooms are

allocated to the timetable periods. In order to do this task, the academic registrar needs to know the number of rooms required, whether the activity is a lecture, tutorial or laboratory, together with the number of students involved. The resourced timetables are then circulated to all members of the academic staff for validation. The final process is to amend the draft timetables in accordance with staff requests, which usually involves moving or swapping timetable entries in order to resolve clashes with other staff commitments. An example of the manually produced Part1 IT degree timetable is shown in Fig 8.3. For example, in the period 0850 to 0940 on Monday a member of staff with initials DCS gives a lecture to module I111, which is common with E101, in room WHLT during weeks 1 to 11.

8.5 Analysis

8.5.1. Ada Package Dependencies

The system software architecture for the timetable production prototype is shown in Fig 8.4; the series of letters at the top of each package symbol indicate Ada package dependencies. The code listings are given in Annex B. In addition, Table 8.3 shows the relationships between the timetable production system given in Fig 8.2 and the Ada packages given in Fig 8.4.

Timetable Production Process	Ada Packages
Determine Module Structure	c
Determine Module Requirement	f
Determine Commonality	d
Allocate Activities & Staff	a, X
Allocate Period	Q, R, S, T, U, Y, W
Allocate Rooms	V
Move or Swap Periods	g,i

Table 8.3 Timetable production process and Ada package relationships.

Time/ Day	0850 0940	0950 1040	1110 1200	1210 1300	1410 1500	1510 1600	1610 1700
Mon	I111(L) DCS (E101) WHLT 1-11	I119(L) WGT (E151) WHLT 1-11	I113(L) BJH MH221 1-11	I113(T) BJH MH221 3,5,7,9	I117(T) HGB JEA SL6B 4,7,9,11 I119(T) WGT SL6B 3,6,8,10 SL6A 9	I118(T) JCM SL6B 2,4,7,10	
Tue		I123 JDP MH223 2,4,6,10 I123(T) JDP MH223 7,11	I114(L) AH (E103) LFLT 1 I116(L) WHTK MH27 2-6, 8-9,11 LFJ LFLT 7 HL14A 10	I114(L) AH (E103) LFLT 1 I116(L) WHTK MH27 2-6, 8-9,11 LFJ LFLT 7 HL14A 10	I122(L) LW MH88 1-4,6-11 MH269 5	I122(T) LW MH88 2,4,7,10 I116(L) LFJ LFLT 9	I116(L) LFJ LFLT 9
Wed	I125(L) DEE WHLT 1,4-7,9,10 LLT 3,11 WH42 2	I111(L) DCS (E101) WHLT 1,3,5,7,9,11 I111(T) RDH 2,4,8,10	I117(L) HGB (E153) HL14A 1-11				
Thu	I115(L) MV MH221 9 JCM MH221 5 I118(L) JCM MH221 1-4,6-8,10	115(L) MV MH221 9 JCM MH221 5 I118(L) JCM MH221 1-4,6-8,10	I114(L) AH (E103) LFLT 1 I114(L) JDP MH27 2-11	I114(L) AH (E103) LFLT 1 I114(L) JDP MH27 5,7,9,11	I115(L) MV MH1 1,3,5,7-11 JCM MH1 2,4,6	I115(T) MV JCM MH1 3,6,8,10 I116 116(L) LFJ LFLT 11	116(L) LFJ LFLT 11
Fri	I111(L) DCS (E101) LFLT 1-5,7,8 WHLT 9-11 LLT 6				I117(P) (E153) HL5A 7,11 I119(P) (E151) HL10A 3,9 I123(P) JDP 10	I117(P) (E153) HL5A 7,11 I119(P) (E151) HL10A 3,9 I123(P) JDP 10	I117(P) (E153) HL5A 7,11 I119(P) (E151) HL10A 3,9 I123(P) JDP 10

Fig 8.3 A manually produced IT degree Part1 timetable.

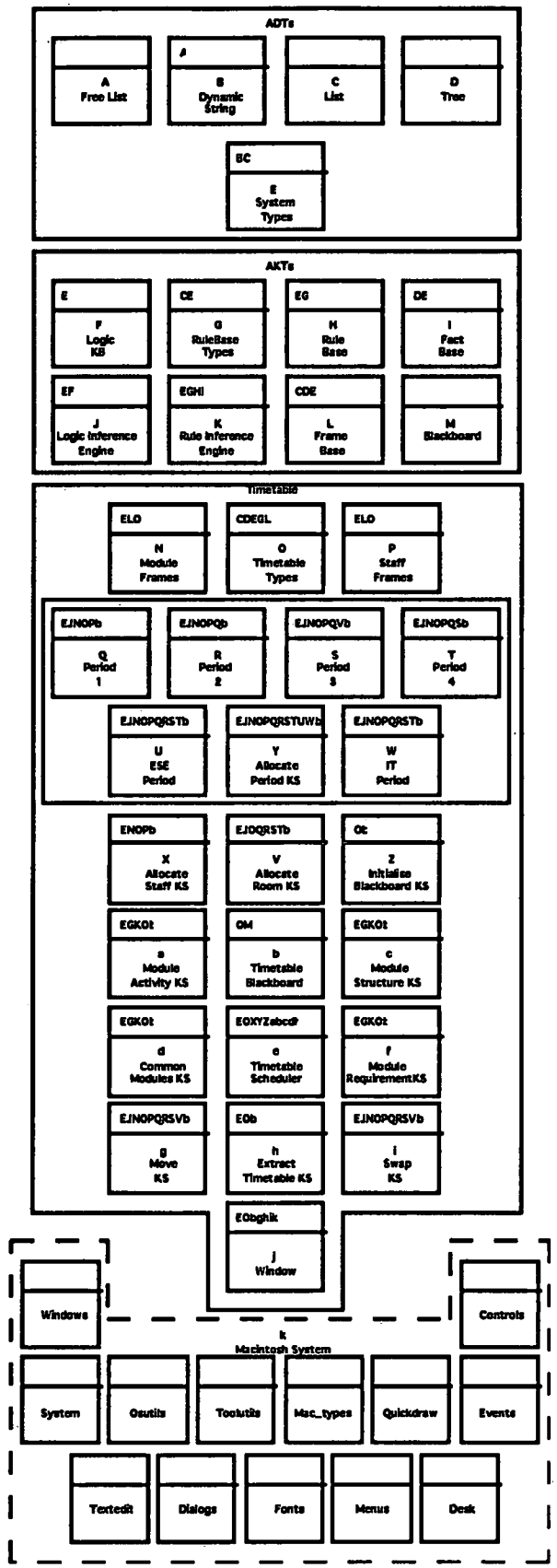

Fig 8.4 Timetable production system software architecture.

8.5.2. Timetable Scheduler

The Timetable Scheduler shown in Fig 8.4 is a simple event list process which selects the appropriate knowledge source to match the current event; an event is caused by a change to the blackboard. Processing is initiated by a call to the Module Structure knowledge source, which reads the module codes from degree files, determines the individual module structure, and places corresponding entries onto the blackboard and event list. The event list entries are then retrieved by the Timetable Scheduler and passed to the appropriate knowledge source for processing.

8.5.3. Blackboard Structure

8.5.3.1 Blackboard Entries

The blackboard entries are application dependent and specified in Timetable Types as:

```
type TIMETABLE_NODE_KIND_TYPE is (
PERIOD_KIND,
MODULE_ACTIVITY_KIND,
MODULE_COMMON_REQUIREMENT_KIND,
MODULE_REQUIREMENT_KIND,
MODULE_KIND);

type TIMETABLE_NODE_RECORD
(KIND : TIMETABLE_NODE_KIND_TYPE := MODULE_ACTIVITY_KIND) is
record
   SUPPORTS      :  TIMETABLE_LIST_PACKAGE.LIST_TYPE;
   ACTION        :  SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACT          :  SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FUSED_FACT    :  SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   RULE_PTR      :  RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;
   SUPPORTERS    :  TIMETABLE_LIST_PACKAGE.LIST_TYPE;
   STUDENTS      :  STANDARD.NATURAL := 0;
   case KIND is
   when PERIOD_KIND                    =>
      PERIOD_DETAIL_FRAMES          :  PERIOD_FRAME_BASE_PACKAGE.
                                       FRAME_BASE_RECORD_TYPE;
   when MODULE_ACTIVITY_KIND           =>
      ACTIVITY                      :  STANDARD.CHARACTER;
```

```
        LIST_OF_COMMON_ACTIVITIES   : SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING_LIST_PACKAGE.
                                      LIST_TYPE;
        STAFF_LIST                  : SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING_LIST_PACKAGE.
                                      LIST_TYPE;
        FREQUENCY                   : SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING.STRING;
        FIRST_PERIOD                : PERIOD_NUMBER_TYPE;
        FIRST_PERIOD_STRING         : STANDARD.STRING(1..2);
        NUMBER_OF_PERIODS           : STANDARD.POSITIVE;
        HAS_PREFERENCE              : STANDARD.BOOLEAN;
        DAY_PREFERENCE              : SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING.STRING;
        PERIOD_PREFERENCE           : SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING.STRING;
    when MODULE_COMMON_REQUIREMENT_KIND =>
        LIST_OF_COMMON_MODULES      : SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING_LIST_PACKAGE.
                                      LIST_TYPE;
        NUMBER_OF_COMMON_MODULES    : STANDARD.POSITIVE := 1;
    when MODULE_REQUIREMENT_KIND        => null;
    when MODULE_KIND                    => null;
    end case;
end record;

type TIMETABLE_NODE_PTR_TYPE is access TIMETABLE_NODE_RECORD;
```

Blackboard entries are allocated as variants of Timetable_Node_Record; note that the variant Period_Kind contains an instance of the frame abstract knowledge type, which illustrates the use of abstract knowledge type instances as entries on the blackboard. In addition, a rule pointer has been included in anticipation of providing a user explanation facility; by following the blackboard entry supporter links the rule that generated each entry can be displayed to provide an explanation of how a particular period entry has been established.

8.5.3.2 Blackboard Structure Specification

Chapter 7 described the construction of an abstract blackboard structure where the user component provides the detailed specification to structure the blackboard levels to meet the application requirement. In the case of the timetable prototype

the blackboard levels are specified as:

```
-- Vertical blackboard divisions

type TIMETABLE_LEVEL_TYPE is(
DAYS,
DEGREE_ACTIVITIES,
COMMON_MODULE_REQUIREMENTS,
MODULE_REQUIREMENTS,
DEGREE_MODULES,
EVENT_LISTS);

-- Horizontal blackboard divisions

type TIMETABLE_ITEM_TYPE is(
MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,
LECTURES, TUTORIALS, CAROUSEL, PRACTICALS,
COMMON_MODULE_REQUIREMENT_LIST,
ESE, IT,
COMMON_EVENTS, PERIOD_EVENTS);
```

The content of each horizontal division is specified as a pointer to a list or an array of blackboard entries as follows:

```
type BLACKBOARD_ITEM_TYPE is (LIST, FRAME);

type BLACKBOARD_ITEM_RECORD(ITEM_KIND : BLACKBOARD_ITEM_TYPE := LIST) is
record
   case ITEM_KIND is
      when LIST    =>  LIST    :  TIMETABLE_LIST_PACKAGE.LIST_TYPE;
      when FRAME   =>  PERIOD  :  PERIOD_ARRAY_TYPE :=
                                  (PERIOD_NUMBER_TYPE =>
                                     new  TIMETABLE_NODE_RECORD
                                          (PERIOD_KIND));
   end case;
end record;

type BLACKBOARD_ITEM_PTR_TYPE is access BLACKBOARD_ITEM_RECORD;
```

The generic instantiation of the timetable blackboard is then:

```
package TIMETABLE_BLACKBOARD is new GENERIC_BLACKBOARD_PACKAGE(
ITEM_TYPE          =>  TIMETABLE_TYPES_PACKAGE.
                       BLACKBOARD_ITEM_RECORD,
ITEM_PTR_TYPE      =>  TIMETABLE_TYPES_PACKAGE.
                       BLACKBOARD_ITEM_PTR_TYPE,
LEVEL_INDEX_TYPE   =>  TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_LEVEL_TYPE,
ITEM_INDEX_TYPE    =>  TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_ITEM_TYPE);
```

which results in the framework shown in Fig 8.5.

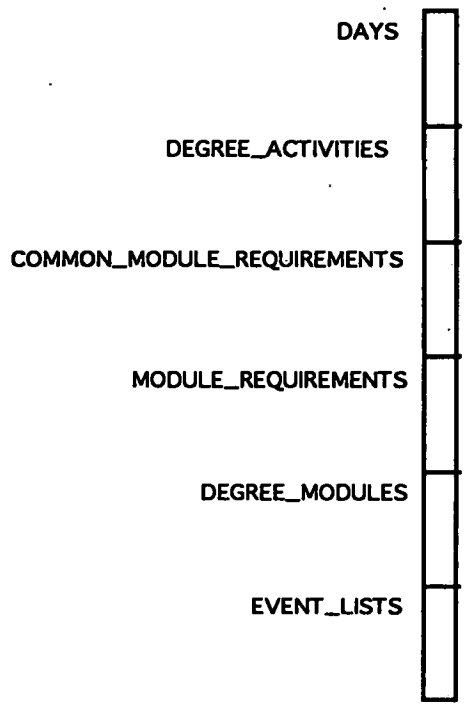

Fig 8.5 Timetable blackboard framework.

8.5.4. Knowledge Sources

8.5.4.1 Initialise Blackboard

The purpose of the Initialise Blackboard knowledge source shown in Fig 8.4 is to construct the empty blackboard architecture prior to processing the module data. The Initialise Blackboard knowledge source is specified as:

```
with   TIMETABLE_TYPES_PACKAGE,
       TIMETABLE_BLACKBOARD_PACKAGE;
package BLACKBOARD_INITIALISE_KS_PACKAGE is
   procedure BUILD_BLACKBOARD_LEVELS(
   BLACKBOARD : in out     TIMETABLE_BLACKBOARD_PACKAGE.
                           TIMETABLE_BLACKBOARD.BLACKBOARD_TYPE);
end BLACKBOARD_INITIALISE_KS_PACKAGE;
```

Procedure Build_Blackboard_Levels receives the blackboard framework shown in Fig 8.5 and allocates the level structures as shown in Fig 8.6 based on the values of the horizontal blackboard divisions given by Timetable_Item_Type.

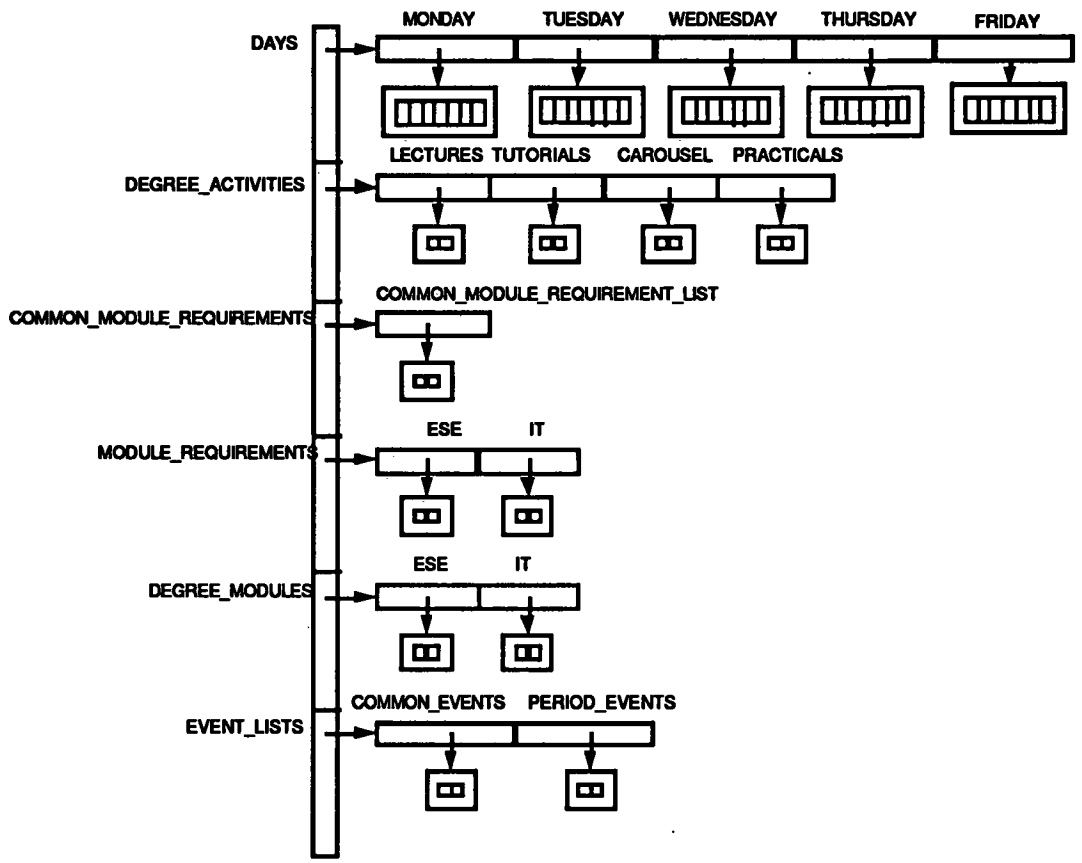

Fig 8.6 The initialised timetable blackboard framework.

Each level is first allocated a constrained array based on Time_Table_Item_Type. Subsequently, each of the array locations on the Days level is allocated a pointer to a blackboard entry with the discriminant Period_Kind; these are not shown in Fig 8.6 in order to reduce complexity. The arrays on all other levels are allocated pointers to the appropriate number of List ADT instances.

8.5.4.2 Module Structure

The Module Structure knowledge source shown in Fig 8.4 carries out the Determine

Module Structure process shown in Fig 8.2 by extracting the module codes from external data files and applying module structure rules to construct the Degree_Modules level of the blackboard. The Module Structure knowledge source is specified as:

```
with   TEXT_IO,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       RULE_BASE_TYPES_PACKAGE,
       GENERIC_RULE_BASE_INFERENCE_PACKAGE,
       TIMETABLE_BLACKBOARD_PACKAGE;
package MODULE_STRUCTURE_KS_PACKAGE is
       procedure GET_DEGREE_MODULES(
       FROM_FILENAME   : in       STANDARD.STRING;
       BLACKBOARD      : In out   TIMETABLE_BLACKBOARD_PACKAGE.
                                  TIMETABLE_BLACKBOARD.BLACKBOARD_TYPE);
end MODULE_STRUCTURE_KS_PACKAGE;
```

The knowledge source integrates an instance of the rule abstract knowledge type by instantiating the generic component in the body of the package.

```
package RULE_BASE_PACKAGE is new
GENERIC_RULE_BASE_INFERENCE_PACKAGE(
RULE_BASES_FILENAME     =>   "p1msubfile.list",
FACT_COMPOSITE_TYPE     =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_RECORD,
FACT_PTR_TYPE           =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE,
"<"                     =>   TIMETABLE_TYPES_PACKAGE."<",
">"                     =>   TIMETABLE_TYPES_PACKAGE.">",
IS_EQUAL                =>   TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
IS_LESS_THAN            =>   TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
IS_GREATER_THAN         =>   TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
PUT                     =>   TIMETABLE_TYPES_PACKAGE.PUT);
```

The file "p1msubfile.list" contains the names of degree files, which in turn contain a list of degree module codes; copies of all knowledge files are given in Annex B. The remaining actual generic parameters define the fact base entries, which are the same as the blackboard entries, and the operations on the entries; these are required by the underlying generic Tree ADT in the fact base. Each file name in "p1msubfile.list" provides the detail for one partition of the rule base.

The rules for Module Structure are formulated to determine the internal structure of a module. For example, E103 is a composite module comprising three sub-components; E103Design, E103Basic and E103Mp which are represented by the simple rule definitions

```
IF    E103
THEN  E103Design

IF    E103
THEN  E103Basic

IF    E103
THEN  E103Mp
```

In this case a single module input code results in three entries on the blackboard Degree_Modules level; an explosive process. Module Structure is called from Timetable Scheduler and each module code results in one or more entries on the Degree_Modules blackboard level; this is illustrated in Fig 8.7.

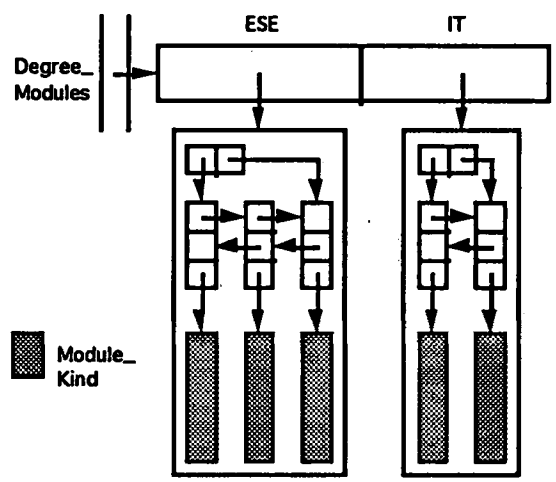

Fig 8.7 Entries on the Degree_Modules blackboard level.

8.5.4.3 Module Requirement

The Module Requirement knowledge source shown in Fig 8.4 performs the Determine Module Requirement process shown in Fig 8.2 by responding to entries

arriving on the Degree_Modules blackboard level. Module Requirement rules are used to generate entries on the Module_Requirements blackboard level. The Module Requirement knowledge source is specified as:

```
with  TEXT_IO,
      SYSTEM_TYPES_PACKAGE,
      TIMETABLE_TYPES_PACKAGE,
      RULE_BASE_TYPES_PACKAGE,
      GENERIC_RULE_BASE_INFERENCE_PACKAGE,
      TIMETABLE_BLACKBOARD_PACKAGE;
package REQUIREMENT_KS_PACKAGE is
   procedure PROCESS_EVENT(
   MODULE_PTR   : in      TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD   : in out  TIMETABLE_BLACKBOARD_PACKAGE.
                          TIMETABLE_BLACKBOARD.
                          BLACKBOARD_TYPE);
end REQUIREMENT_KS_PACKAGE;
```

An instance of the rule abstract knowledge type is integrated into Module Requirement by the single instantiation of the generic rule component:

```
package RULE_BASE_PACKAGE is new
GENERIC_RULE_BASE_INFERENCE_PACKAGE(
RULE_BASES_FILENAME     =>   "p1mreqfile.list",
FACT_COMPOSITE_TYPE     =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_RECORD,
FACT_PTR_TYPE           =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE,
"<"                     =>   TIMETABLE_TYPES_PACKAGE."<",
">"                     =>   TIMETABLE_TYPES_PACKAGE.">",
IS_EQUAL                =>   TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
IS_LESS_THAN            =>   TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
IS_GREATER_THAN         =>   TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
PUT                     =>   TIMETABLE_TYPES_PACKAGE.PUT);
```

The file "p1mreqfile.list" contains the file names of degree files, which in turn contain the module requirement rules. Like the previous instantiation, the remaining actual generic parameters define the fact base entries, which in this case are the entries on the Degree_Modules level.

The rules for Module Requirement are formulated to determine the lecture, tutorial, practical and carousel requirements for each module. For example, the

rule

IF E151
THEN provide_lecture_tutorial_carousel

states that if the current event is a blackboard entry for E151, then the module requires lecture, tutorial and carousel resources; the carousel is a timetable of laboratory sessions programmed separately to the main timetable. The Module Requirement knowledge source interprets the action part of the rule in order to establish the module requirement.

Module Requirement is called from Timetable Scheduler on intercepting an event placed on the event list by Module Structure. The result of the knowledge source action is to place an entry on the Module_Requirements level, and to connect the entries by adding elements to the supports and supporters lists in the Degree Modules and Module Requirements entries respectively. An example of the resulting blackboard structure is shown in Fig 8.8.

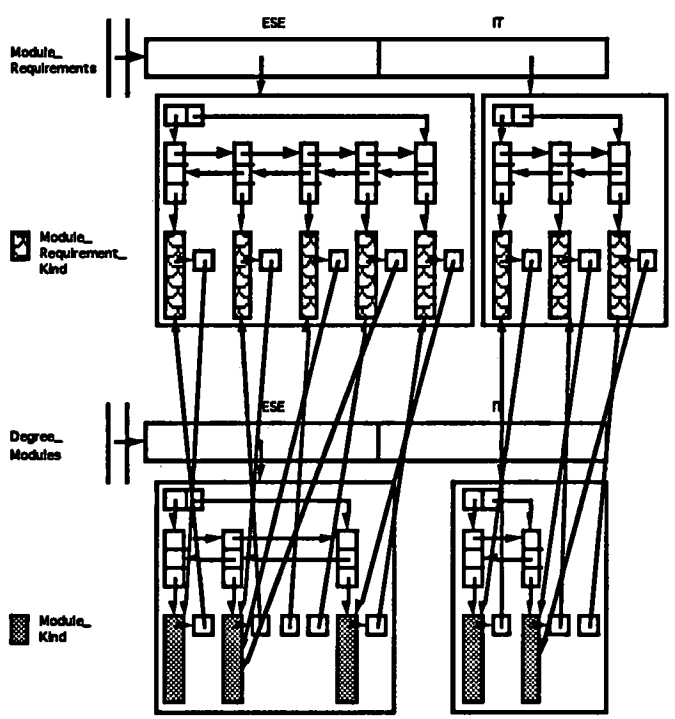

Fig 8.8 Module_Requirements blackboard level.

Note how some of the entries on the Degree_Modules level expand to two or more entries on the Module_Requirements level to meet the lecture, tutorial, carousel and/or practical requirement of each module; this is an another example of an explosive process.

8.5.4.4 Common Modules

The purpose of the Common Modules knowledge source shown in Fig 8.4 is to carry out the Determine Commonality process shown in Fig 8.2 by reacting to entries arriving on the Module Requirements blackboard level and by using the common module rules to generate entries on the Common_Module_Requirements level. The Common Modules knowledge source is specified as:

```
with   TEXT_IO,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       RULE_BASE_TYPES_PACKAGE,
       GENERIC_RULE_BASE_INFERENCE_PACKAGE,
       TIMETABLE_BLACKBOARD_PACKAGE;
package COMMON_KS_PACKAGE is
   procedure PROCESS_EVENT(
   MODULE_REQUIREMENT_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                                       TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD                : in out  TIMETABLE_BLACKBOARD_PACKAGE.
                                       TIMETABLE_BLACKBOARD.
                                       BLACKBOARD_TYPE);
end COMMON_KS_PACKAGE;


package RULE_BASE_PACKAGE is new
GENERIC_RULE_BASE_INFERENCE_PACKAGE(
RULE_BASES_FILENAME     =>   "p1mcomfile.list",
FACT_COMPOSITE_TYPE     =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_RECORD,
FACT_PTR_TYPE           =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE,
"<"                     =>   TIMETABLE_TYPES_PACKAGE."<",
">"                     =>   TIMETABLE_TYPES_PACKAGE.">",
IS_EQUAL                =>   TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
IS_LESS_THAN            =>   TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
IS_GREATER_THAN         =>   TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
PUT                     =>   TIMETABLE_TYPES_PACKAGE.PUT);
```

A single instance of the rule abstract knowledge type is integrated into the knowledge source as indicated in the above instantiation. The file "p1mcomfile.list" contains the names of degree commonality files, which in turn contain the module commonality rules.

The rules for Common Modules are formulated to determine the relationship between different degree modules. For example,

```
IF    I111(L) AND E101(L)
THEN make_I111(L)_and_E101(L)_common
```

shows that if the events I111(L) and E101(L) have occurred then they are to be made common, which results in the knowledge source generating one entry on the Common_Module_Requirements level to represent the two entries on the Module_Requirements level; this is a fusion process and is illustrated in Fig 8.9.

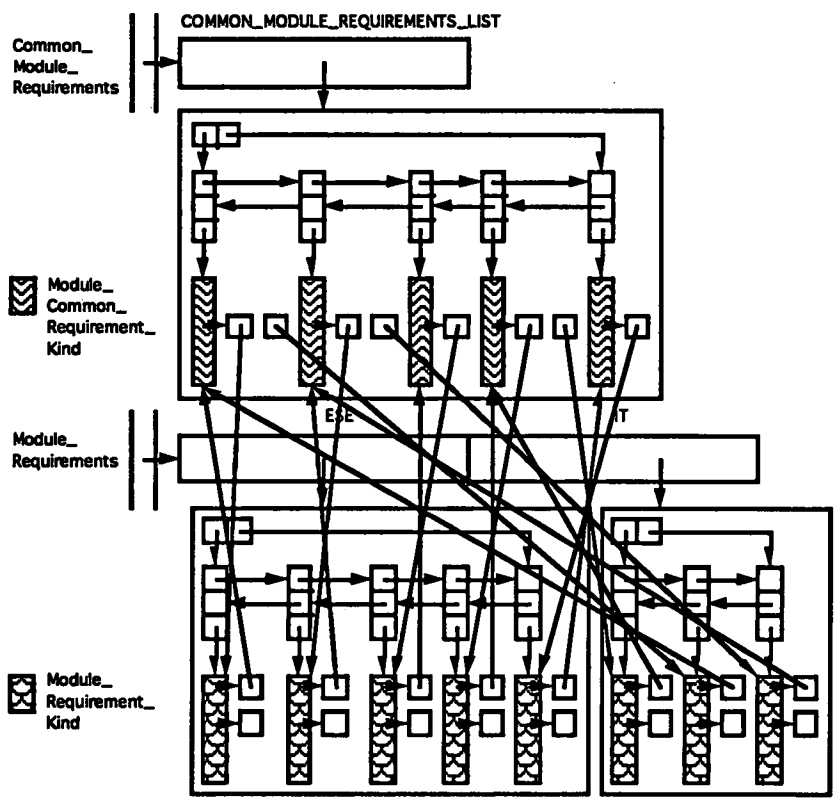

Fig 8.9 The Common_Module_Requirements blackboard level.

8.5.4.5 Module Activity

The Module_Activity knowledge source, shown in Fig 8.4, allocates activities by reacting to entries arriving on the Common_Module_Requirement blackboard level and by using module activity rules to generate entries on the Degree_Activities blackboard level. The Module Activity knowledge source is specified as:

```
with   TEXT_IO,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       RULE_BASE_TYPES_PACKAGE,
       GENERIC_RULE_BASE_INFERENCE_PACKAGE,
       TIMETABLE_BLACKBOARD_PACKAGE;
package ACTIVITY_KS_PACKAGE is
   procedure PROCESS_EVENT(
   COMMON_MODULE_PTR  : in       TIMETABLE_TYPES_PACKAGE.
                                 TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD         : in out   TIMETABLE_BLACKBOARD_PACKAGE.
                                 TIMETABLE_BLACKBOARD.
                                 BLACKBOARD_TYPE);
end ACTIVITY_KS_PACKAGE;
```

Like the two previous knowledge sources, Module Activity integrates an instance of the rule abstract knowledge type. This is achieved by the single instantiation:

```
package RULE_BASE_PACKAGE is new
GENERIC_RULE_BASE_INFERENCE_PACKAGE(
RULE_BASES_FILENAME     =>   "p1mactfile.list",
FACT_COMPOSITE_TYPE     =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_RECORD,
FACT_PTR_TYPE           =>   TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE,
"<"                     =>   TIMETABLE_TYPES_PACKAGE."<",
">"                     =    TIMETABLE_TYPES_PACKAGE.">",
IS_EQUAL                =>   TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
IS_LESS_THAN            =>   TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
IS_GREATER_THAN         =>   TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
PUT                     =>   TIMETABLE_TYPES_PACKAGE.PUT);
```

The file "p1mactfile.list" contains the names of degree files, which in turn contain rules to determine the module activities; the remaining actual generic parameters define the fact base and its operations, in this case the facts are on the Common_Module_Requirements blackboard level.

The rules for Module Activity are formulated to determine the number of lectures, tutorials and practicals that a module requires. For example, the rules:

```
IF    I123(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3,5,8,10

IF    E103Basic(P)
THEN allocate_1_/_2_period_Practical_in_weeks_2-5,7-10_THU-3
```

show how the number of periods, type of activity, weeks required and day preference are implemented. In the first rule, one single tutorial period in weeks 3,5,8 and 10 is required. In the second rule, one double practical period is required in weeks 2 to 5 and 7 to 10, with a preference starting Thursday period 3.

Module Activity is called from Timetable Scheduler after the detection of a Common_Module_Requirements event. The Module Activity action results in the addition of appropriate activities to the Degree_Activities level; note that this is another explosive process which is illustrated in Fig 8.10.

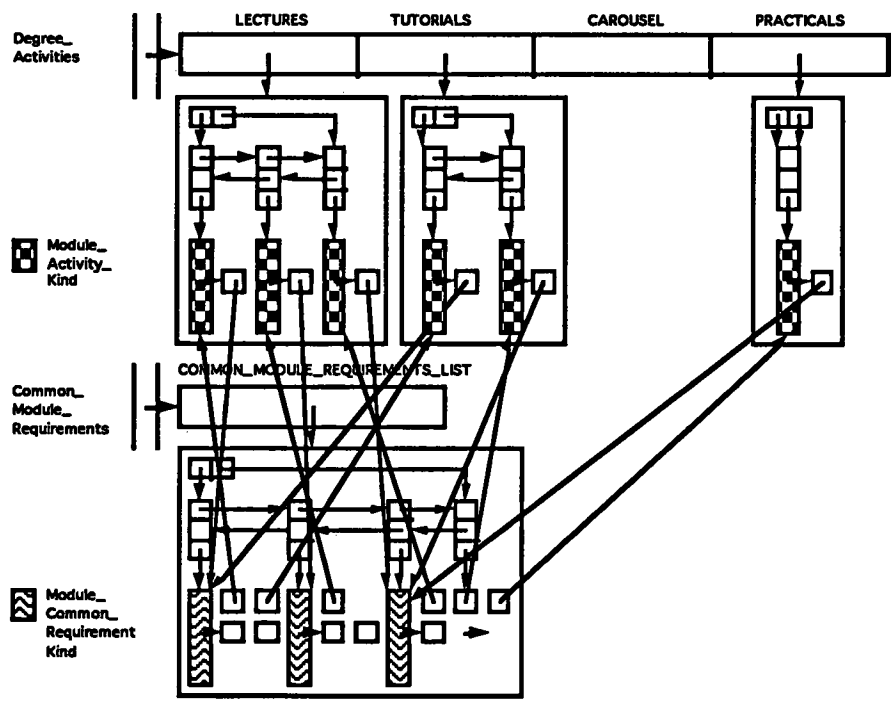

Fig 8.10 The Degree Activities level.

8.5.4.6 Allocate Staff

Allocate Staff is the knowledge source responsible for selecting the appropriate staff member to resource a particular activity that has been entered on the Degree_Activities blackboard level.

8.5.4.6.1 Staff and Module Frames

During the design of the Allocate Staff knowledge source it became clear that access to staff/module details would be required in more than one part of the timetable production process; for example, in the initial staff allocation process and in the Swap and Move period knowledge sources. Consequently, independent instantiations of the frame abstract knowledge type were constructed for staff and module data. Staff Frames is specified as:

```
with   GENERIC_FRAME_BASE_PACKAGE,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       TEXT_IO;
package STAFF_PACKAGE is
   package STAFF_FRAME_BASE_PACKAGE is new GENERIC_FRAME_BASE_PACKAGE(
   SLOT_TYPE                    =>   TIMETABLE_TYPES_PACKAGE.
                                     STAFF_SLOT_TYPE,
   FACET_RECORD_TYPE            =>   TIMETABLE_TYPES_PACKAGE.
                                     STAFF_FACET_RECORD_TYPE,
   FACET_RECORD_PTR_TYPE        =>   TIMETABLE_TYPES_PACKAGE.
                                     STAFF_FACET_RECORD_PTR_TYPE,
   "<"                          =>   TIMETABLE_TYPES_PACKAGE."<",
   ">"                          =>   TIMETABLE_TYPES_PACKAGE.">",
   IS_EQUAL                     =>   TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
   IS_LESS_THAN                 =>   TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
   IS_GREATER_THAN              =>   TIMETABLE_TYPES_PACKAGE.
                                     IS_GREATER_THAN,
   PUT                          =>   TIMETABLE_TYPES_PACKAGE.PUT,
   GET                          =>   TIMETABLE_TYPES_PACKAGE.GET);

   STAFF_FRAME_BASE_RECORD  :        STAFF_FRAME_BASE_PACKAGE.
                                     FRAME_BASE_RECORD_TYPE;

   procedure MAKE_BUSY(
   LECTURER                     :  in       SYSTEM_TYPES_PACKAGE.
```

```
                                                    DYNAMIC_STRING.STRING;
MODULE                          : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
DAY                             : in                TIMETABLE_TYPES_PACKAGE.
                                                    DAY_TYPE;
PERIOD                          : in                TIMETABLE_TYPES_PACKAGE.
                                                    PERIOD_NUMBER_TYPE;
WEEK_ARRAY                      : in                TIMETABLE_TYPES_PACKAGE.
                                                    WEEK_ARRAY_TYPE;
STAFF_FRAME_BASE_RECORD         : in out            STAFF_FRAME_BASE_PACKAGE.
                                                    FRAME_BASE_RECORD_TYPE);

procedure MAKE_FREE(
LECTURER                        : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
MODULE                          : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
DAY                             : in                TIMETABLE_TYPES_PACKAGE.
                                                    DAY_TYPE;
PERIOD                          : in                TIMETABLE_TYPES_PACKAGE.
                                                    PERIOD_NUMBER_TYPE;
WEEK_ARRAY                      : in                TIMETABLE_TYPES_PACKAGE.
                                                    WEEK_ARRAY_TYPE;
STAFF_FRAME_BASE_RECORD         : in out            STAFF_FRAME_BASE_PACKAGE.
                                                    FRAME_BASE_RECORD_TYPE);

function IS_FREE(
LECTURER                        : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
DAY                             : in                TIMETABLE_TYPES_PACKAGE.
                                                    DAY_TYPE;
PERIOD                          : in                TIMETABLE_TYPES_PACKAGE.
                                                    PERIOD_NUMBER_TYPE;
WEEK_ARRAY                      : in                TIMETABLE_TYPES_PACKAGE.
                                                    WEEK_ARRAY_TYPE;
STAFF_FRAME_BASE_RECORD         : in                STAFF_FRAME_BASE_PACKAGE.
                                                    FRAME_BASE_RECORD_TYPE)
return BOOLEAN;

procedure MAKE_ALL_STAFF_BUSY(
STAFF_LIST                      : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING_LIST_PACKAGE.
                                                    LIST_TYPE;
MODULE                          : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
DAY                             : in                TIMETABLE_TYPES_PACKAGE.
                                                    DAY_TYPE;
PERIOD                          : in                TIMETABLE_TYPES_PACKAGE.
                                                    PERIOD_NUMBER_TYPE;
WEEK_ARRAY                      : in                TIMETABLE_TYPES_PACKAGE.
                                                    WEEK_ARRAY_TYPE;
STAFF_FRAME_BASE_RECORD         : in out            STAFF_FRAME_BASE_PACKAGE.
                                                    FRAME_BASE_RECORD_TYPE);
procedure MAKE_ALL_STAFF_FREE(
STAFF_LIST                      : in                SYSTEM_TYPES_PACKAGE.
```

```
                                                    DYNAMIC_STRING_LIST_PACKAGE.
                                                    LIST_TYPE;
    MODULE                      : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
    DAY                         : in                TIMETABLE_TYPES_PACKAGE.
                                                    DAY_TYPE;
    PERIOD                      : in                TIMETABLE_TYPES_PACKAGE.
                                                    PERIOD_NUMBER_TYPE;
    WEEK_ARRAY                  : in                TIMETABLE_TYPES_PACKAGE.
                                                    WEEK_ARRAY_TYPE;
    STAFF_FRAME_BASE_RECORD     : in out            STAFF_FRAME_BASE_PACKAGE.
                                                    FRAME_BASE_RECORD_TYPE);

    function ALL_STAFF_FREE(
    STAFF_LIST                  : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING_LIST_PACKAGE.
                                                    LIST_TYPE;
    MODULE                      : in                SYSTEM_TYPES_PACKAGE.
                                                    DYNAMIC_STRING.STRING;
    DAY                         : in                TIMETABLE_TYPES_PACKAGE.
                                                    DAY_TYPE;
    PERIOD                      : in                TIMETABLE_TYPES_PACKAGE.
                                                    PERIOD_NUMBER_TYPE;
    WEEK_ARRAY                  : in                TIMETABLE_TYPES_PACKAGE.
                                                    WEEK_ARRAY_TYPE;
    STAFF_FRAME_BASE_RECORD     : in                STAFF_FRAME_BASE_PACKAGE.
                                                    FRAME_BASE_RECORD_TYPE)

    return BOOLEAN;
end STAFF_PACKAGE;
```

The module staff requirements were placed in Module Frames. The specification of Module Frames is much simpler since it was anticipated that this frame base would be read only. Consequently, Module Frames is specified as:

```
with   GENERIC_FRAME_BASE_PACKAGE,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       TEXT_IO;
package MODULE_PACKAGE is

    package MODULE_FRAME_BASE_PACKAGE is new GENERIC_FRAME_BASE_PACKAGE
    (SLOT_TYPE                  =>  TIMETABLE_TYPES_PACKAGE.
                                    MODULE_SLOT_TYPE,
    FACET_RECORD_TYPE           =>  TIMETABLE_TYPES_PACKAGE.
                                    MODULE_FACET_RECORD_TYPE,
    FACET_RECORD_PTR_TYPE       =>  TIMETABLE_TYPES_PACKAGE.
                                    MODULE_FACET_RECORD_PTR_TYPE,
    "<"                         =>  TIMETABLE_TYPES_PACKAGE."<",
    ">"                         =>  TIMETABLE_TYPES_PACKAGE.">",
    IS_EQUAL                    =>  TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
```

```
      IS_LESS_THAN                  =>  TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
      IS_GREATER_THAN               =>  TIMETABLE_TYPES_PACKAGE.
                                        IS_GREATER_THAN,
      PUT                           =>  TIMETABLE_TYPES_PACKAGE.PUT,
      GET                           =>  TIMETABLE_TYPES_PACKAGE.GET);

      MODULE_FRAME_BASE_RECORD   :  MODULE_FRAME_BASE_PACKAGE.
                                    FRAME_BASE_RECORD_TYPE;

      procedure FIND_STAFF(
      MODULE                     :  in       SYSTEM_TYPES_PACKAGE.
                                             DYNAMIC_STRING.STRING;
      STAFF_SLOT_PTR             :  in out   MODULE_FRAME_BASE_PACKAGE.
                                             SLOT_PACKAGE.
                                             SLOT_RECORD_PTR_TYPE;
      MODULE_FRAME_BASE_RECORD   :  in       MODULE_FRAME_BASE_PACKAGE.
                                             FRAME_BASE_RECORD_TYPE);
end MODULE_PACKAGE;
```

8.5.4.6.2 Allocate Staff Knowledge Source

The Allocate Staff knowledge source shown in Fig 8.4 is specified as:

```
with  TEXT_IO,
      SYSTEM_TYPES_PACKAGE,
      STAFF_PACKAGE,
      MODULE_PACKAGE,
      TIMETABLE_TYPES_PACKAGE,
      TIMETABLE_BLACKBOARD_PACKAGE;
package STAFF_KS_PACKAGE is
   procedure PROCESS_EVENT(
   ACTIVITY_PTR     :  in       TIMETABLE_TYPES_PACKAGE.
                                TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD       :  in out   TIMETABLE_BLACKBOARD_PACKAGE.
                                TIMETABLE_BLACKBOARD.
                                BLACKBOARD_TYPE);
end STAFF_KS_PACKAGE;
```

The Allocate Staff knowledge source uses the external staff and module frame bases and does not integrate any new abstract knowledge type instance. The knowledge source is called from Timetable Scheduler, which is triggered by an event on the Degree_Activities blackboard level produced by Module_Activity. The action of the knowledge source is to consult the module frame base to determine the staff

member(s) responsible for the module causing the current event. The staff member names are recorded in the blackboard entry causing the event, and staff availability updated in the staff frame base. Consequently, this knowledge source causes a change to an existing entry on the blackboard rather than creating a new entry.

8.5.4.6.3 Allocate Period

The Allocate Period knowledge source is the most complicated of the knowledge sources and uses the logic abstract knowledge type to allocate period resources to the activities arriving on the Degree_Activities blackboard level. Furthermore, as the module grew, it had to be split into seven separate packages to reduce complexity and make it easier to use; the packages are Period1-4, ESE Period, IT Period and Allocate Period knowledge source shown in Fig 8.4.

Each degree timetable is specified as a PROLOG knowledge base by creating separate instances of the logic abstract knowledge type. The degree timetable knowledge bases are integrated by the simple instantiations:

```
package ESE_PERIOD_INFERENCE_PACKAGE is new LOGIC_INFERENCE_PACKAGE("ESE");
package IT_PERIOD_INFERENCE_PACKAGE is new LOGIC_INFERENCE_PACKAGE("IT");
```

The knowledge bases are then constructed using the logic Build operations:

```
ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
BUILD(
KB          =>    ESE_PERIOD_KB,
FILE_NAME   =>    "ESEperiod.pro");

IT_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
BUILD(
KB          =>    IT_PERIOD_KB,
FILE_NAME   =>    "ITperiod.pro");
```

This is a simple illustration of multiple abstract knowledge type instances used in a single knowledge source; in a fully operational system there would be a logic instance for each degree. The files "ESEperiod.pro" and ITperiod.pro" contain the PROLOG representations for each degree timetable. Each period is defined by a PROLOG fact, for example:

```
period(mon, p1, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p2, wks(o,x,o,x,o,x,o,x,o,x,o)).
```

where the first fact is used to record that the first period on each Monday for the eleven weeks of the term is free, while the second fact signifies that the second period on each Tuesday is free in weeks 1,3,5,7,9 and 11, but occupied in weeks 2,4,6,8 and 10.

The allocation of periods is defined by PROLOG rules, for example:

```
three_periods(D, P1, P2, P3,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11),
wks(P3W1,P3W2,P3W3,P3W4,P3W5,P3W6,P3W7,P3W8,P3W9,P3W10,P3W11)) :-
period(D,p5,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
      P1W11)),
period(D,p6,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
      P2W11)),
period(D,p7,wks(P3W1,P3W2,P3W3,P3W4,P3W5,P3W6,P3W7,P3W8,P3W9,P3W10,
      P3W11)),
P1 is p5,
P2 is p6,
P3 is p7.
```

is used to request three consecutive periods on a day represented by the PROLOG variable D. Note that the three period allocator rule is restricted to periods 5, 6 and 7; the member of staff responsible for constructing the timetable decided that three consecutive periods are best programmed in the afternoon since the four morning periods are broken in the middle by a thirty minute break.

Allocate Period is called from Timetable Scheduler in response to a Degree_Activities event on completion of staff allocation. The result of this action is to add an entry to the Days blackboard level; this materialises as an addition of a frame and associated slot and facets to the selected day/period frame base. The Days blackboard level is illustrated in Fig 8.11.

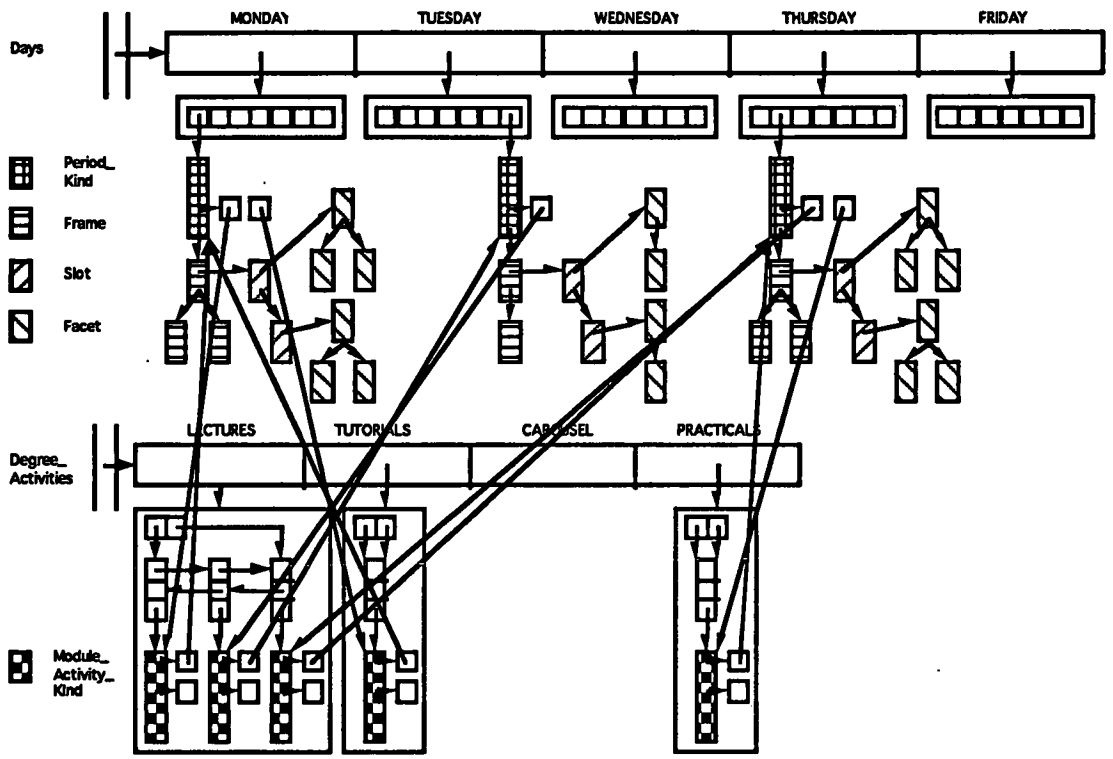

Fig 8.11 The Days blackboard level.

A fragment of the frame structure for the period on Tuesday at 1110 to 1200, shown in Fig 8.13, is given for illustration in Fig 8.12.

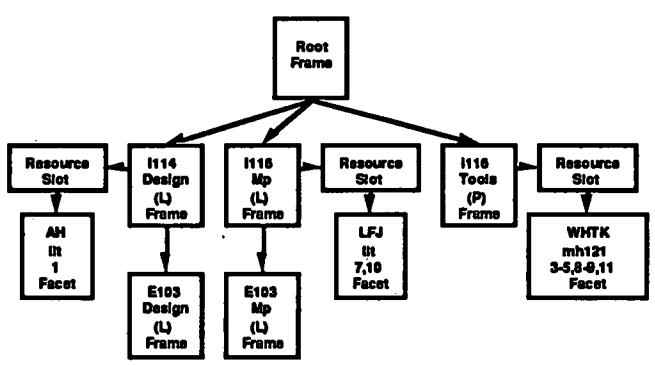

Fig 8.12 A fragment of a period frame base.

8.5.4.6.4 Allocate Rooms

As soon as the period for any given activity has been established Allocate Rooms is called to select the most appropriate venue. The Allocate Rooms knowledge source is specified as:

```
with   TEXT_IO,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       TIMETABLE_BLACKBOARD_PACKAGE,
       LOGIC_INFERENCE_PACKAGE,
       PERIOD_1_PACKAGE,
       PERIOD_2_PACKAGE,
       PERIOD_3_PACKAGE,
       PERIOD_4_PACKAGE;
package ROOM_PACKAGE is

   procedure RE_ALLOCATE_ROOM(
      PERIOD_PTR    :  TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
      ACTIVITY_PTR :   TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
      DAY           :  TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_ITEM_TYPE;
      PERIOD        :  TIMETABLE_TYPES_PACKAGE.
                       PERIOD_NUMBER_TYPE);

   procedure ALLOCATE_ROOMS(
   ROOMS             : in out    SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING_ARRAY;
   PERIODS           : in        SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING_ARRAY;
   THIS_DAY          : in        TIMETABLE_TYPES_PACKAGE.
                                 DAY_TYPE;
   WEEKS_REQUIRED    : in        TIMETABLE_TYPES_PACKAGE.
                                 WEEK_ARRAY_TYPE;
   ACTIVITY_PTR      : in        TIMETABLE_TYPES_PACKAGE.
                                 TIMETABLE_NODE_PTR_TYPE;
   SELECTED          :      out  BOOLEAN);
end ROOM_PACKAGE;
```

Allocate Rooms also integrates a logic abstract knowledge type. This is achieved by the single generic instantiation

```
package ROOM_INFERENCE_PACKAGE is new LOGIC_INFERENCE_PACKAGE("ROOM");
```

and subsequently built using

```
ROOM_INFERENCE_PACKAGE.
LOGIC_KB.
BUILD(
KB          =>    ROOM_KB,
FILE_NAME   =>    "room.pro");
```

The file "room.pro" provides the PROLOG knowledge base with facts and rules to represent room usage across the timetable. For example, the fact

```
room(mh121, 40, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o)).
```

represents the room mh121, which seats 40, period 1 on Monday for the eleven weeks of a term. The state of this fact signifies that mh121 is available in period 1 for all eleven weeks. The facts are retracted and re-asserted as rooms are allocated. For example,

```
room(mh121, 40, mon, p1, w(x,o,x,o,x,o,x,o,x,o,x)).
```

represents room mh121 occupied in weeks 1, 3, 5, 7, 9 and 11.

An example of a room allocation rule is:

```
room_periods_2(
   R1, S1, D1, P1, P2,
   (W1, W2, W3, W4, W5, W6, W7, W8, W9, W10, W11)) :-
   room(R1, C1, D1, P1, w(W1, W2, W3, W4, W5, W6, W7, W8, W9, W10, W11)),
   S1 < C1,
   room(R1, C1, D1, P2, w(W1, W2, W3, W4, W5, W6, W7, W8, W9, W10, W11)).
```

This rule is used to request any room, R1, for use in periods P1 and P2 on day D1 with S1 students.

Allocate room is called from Allocate Period at the point the period allocation is made, and updates the content of existing Days blackboard level entries. Re_Allocate room is called when moving or swapping periods in the timetable.

8.5.4.7 Move Period

The Move Period knowledge source responds to requests from a user to move a module that has already been allocated to a period that Is vacant. The Move Period knowledge source is specified as:

```
with  TEXT_IO,
      SYSTEM_TYPES_PACKAGE,
      TIMETABLE_TYPES_PACKAGE,
      TIMETABLE_BLACKBOARD_PACKAGE,
      STAFF_PACKAGE,
      MODULE_PACKAGE,
      LOGIC_INFERENCE_PACKAGE,
      PERIOD_1_PACKAGE,
      PERIOD_2_PACKAGE,
      PERIOD_3_PACKAGE,
      PERIOD_4_PACKAGE,
      PERIOD_ESE_PACKAGE,
      PERIOD_IT_PACKAGE;
package MOVE_PACKAGE is
   procedure MOVE_TO(
   FM_ACTIVITY_NAME  : in      STANDARD.STRING;
   FM_DAY            : in      TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_ITEM_TYPE;
   FM_PERIOD         : in      TIMETABLE_TYPES_PACKAGE.
                               PERIOD_NUMBER_TYPE;
   TO_DAY            : in      TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_ITEM_TYPE;
   TO_PERIOD         : in      TIMETABLE_TYPES_PACKAGE.
                               PERIOD_NUMBER_TYPE;
   BLACKBOARD        : in      TIMETABLE_BLACKBOARD_PACKAGE.
                               TIMETABLE_BLACKBOARD.
                               BLACKBOARD_TYPE;
   SUCCESS           :    out  BOOLEAN);
end MOVE_PACKAGE;
```

No new abstract knowledge types are integrated into Move Period since all the knowledge has already been created. The single operation Move_To uses the existing logic instances for the degree timetables, and the frame instances for staff and modules, to determine whether a requested move of a module from one period to another is feasible. If all abstract knowledge type instances agree to the move, then the request is actioned by moving the module frames from their current location on

the blackboard Day level to the requested destination day/period. However, If the degree period logic instances find that the day/period to which the move has been requested is already occupied, then the action is denied. Furthermore, if the subsequent staff move is not possible, due to other commitments, the move request is also denied. Finally, a room is requested through the room logic abstract knowledge type instance.

8.5.4.8 Swap Periods

The Swap Periods knowledge source responds to a request to swap one module in one period with a module in another period. Swap Periods is specified as:

```
with   TEXT_IO,
       SYSTEM_TYPES_PACKAGE,
       TIMETABLE_TYPES_PACKAGE,
       TIMETABLE_BLACKBOARD_PACKAGE,
       LOGIC_INFERENCE_PACKAGE,
       STAFF_PACKAGE,
       PERIOD_1_PACKAGE,
       PERIOD_2_PACKAGE,
       PERIOD_3_PACKAGE,
       PERIOD_4_PACKAGE,
       MOVE_PACKAGE;
package SWAP_PACKAGE is
   procedure TRY_SWAPPING_PERIODS(
   FM_ACTIVITY_NAME  : in      STANDARD.STRING;
   FM_DAY            : in      TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_ITEM_TYPE;
   FM_PERIOD         : in      TIMETABLE_TYPES_PACKAGE.
                               PERIOD_NUMBER_TYPE;
   TO_ACTIVITY_NAME  : in      STANDARD.STRING;
   TO_DAY            : in      TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_ITEM_TYPE;
   TO_PERIOD         : in      TIMETABLE_TYPES_PACKAGE.
                               PERIOD_NUMBER_TYPE;
   BLACKBOARD        : in      TIMETABLE_BLACKBOARD_PACKAGE.
                               TIMETABLE_BLACKBOARD.
                               BLACKBOARD_TYPE;
   SUCCESS           :    out  BOOLEAN);
end SWAP_PACKAGE;
```

Again, no new abstract knowledge type is needed. The action of this knowledge source is similar to that of Move, except that two modules are identified to be

swapped in the timetable.

8.5.4.9 Extract Timetable

Extract Timetable is a procedural knowledge source which extracts the individual timetables from the Days blackboard level. The information for each degree timetable on any particular day/period is stored in the frame abstract knowledge type instance attached to that particular day/period. Extract Timetable scans the blackboard, interrogating each of the frame bases to extract the detail. As the detail in each frame is retrieved, it is added to the appropriate degree timetable.

8.5.4.10 Window

Although the implementation of the Window knowledge source consumed a disproportionate amount of time, due to the very poor Ada to Macintosh system mapping documentation, the experience did not make any contribution to the overall research. Consequently, no detailed description is given here, but the complete code specification and implementation are given in Annex B.

One of the outputs of the Window knowledge source is the timetable illustrated in Fig 8.13. In addition, menus are provided to allow the user to select the degree that is to be displayed; the menus are shown in Fig 8.14. Furthermore, on selecting the Move or Swap period options interactive windows are presented to the user in order to obtain the days, periods and module names involved in the selected operation; the windows are shown in Fig 8.15.

Time / Day	0850 0940	0950 1040	1110 1200	1210 1300	1410 1500	1510 1600	1610 1700
Mon	I111(L) E101(L) DCS 1-11 lit	I113(L) BJH 1-11 mh121	I111(L) E101(L) DCS 1-11 mh169	I114Prolog(P) JDP 5,7,9,11 mh121	I115(L) MLV 1-11 mh121	I117(T) HGB 4,7,9,11 mh121 JEA 4,7,9,11 mh223 I123(T) JDP 3,5,8,10 mh223	I119(T) WGT 3,6,8-10 mh121
Tue	I115(L) MLV 5,9 mh121 I118(T) JCM 2,4,7,10 mh223	I115(L) MLV 5,9 mh121 I115(T) MLV3,6,8,10 mh121	I114Design(L) E103Design(L) AH 1 lit I116Mp(L) E103Mp(L) LFJ 7,10 lit I116Tools(P) WHTK 3-5,8,9,11 mh121	I114Design(L) E103Design(L) AH 1 lit I116Mp(L) E103Mp(L) LFJ 7,10 lit I116Tools(P) WHTK 2-6,8,9,11 mh121	I122(L) LW 1-11 mh121	I122(T) LW 2,4,7,10 mh121	I123(L) JDP 1-11 mh121
Wed	I125(L) DEE 1-11 mh121	I111(L) E101(L) DCS 1,3,5,7,9,11 lit	I119(L) E151(L) WGT 1-11 mh169				
Thu	I117(L) E153(L) HGB 1-11 mh169		I114Design(L) E103Design(L) AH 1 lit I114Prolog(T) JDP 2-11 mh121	I114Design(L) E103Design(L) AH 1 lit I111(T) DCS 2,4,6,8,10 mh121 I113(T) BJH 3,5,7,9 mh121	I123(P) JDP 10 mh121	I123(P) JDP 10 mh121	I123(P) JDP 10 mh121
Fri	I118(L) JCM 1-4,6-8,10 mh121	I118(L) JCM 1-4,6-8,10 mh121					

Fig 8.13 The automatically produced IT degree Part1 timetable.

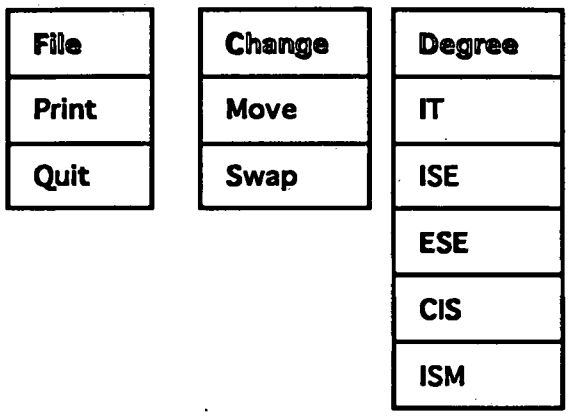

Fig 8.14 The timetable menus.

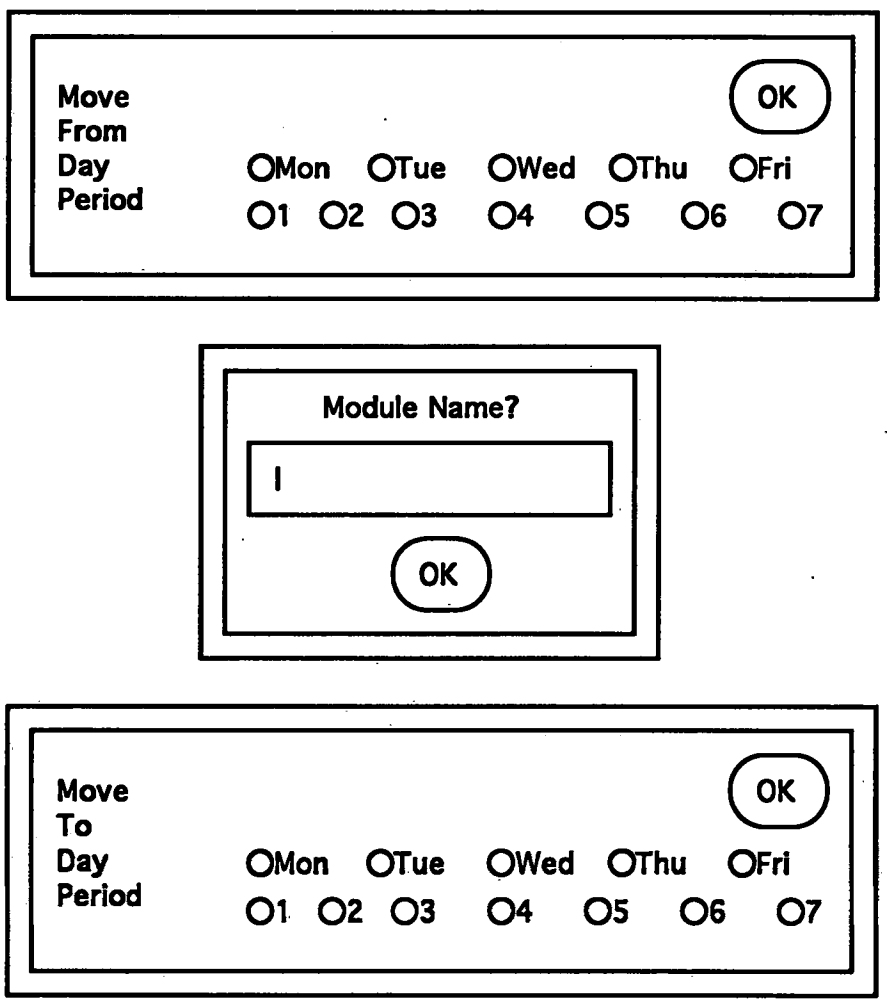

Fig 8.15 The timetable interactive windows.

8.6 Results

The test results are recorded in the following table and charts. Only those modules that recorded significant CPU time consumption are shown; all other modules recorded less than 3.3%. Discussion of these results is deferred to Chapter 9.

Item	Lines of Code	CPU Time %	Ada new
Dynamic String	828	72.9	23087
Logic KB	1822	6.1	9838
Allocate Room KS	837	4.5	21462
Min:Sec		0:12.21	

Table 8.4 Timetable production results.

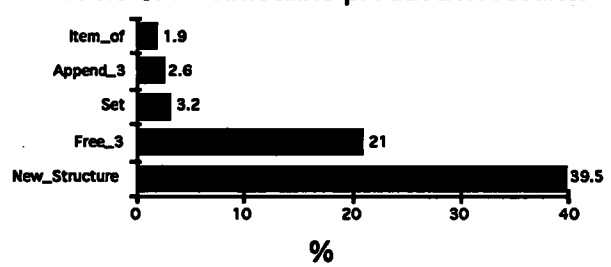

Chart 8.1 Timetable production dynamic string analysis.

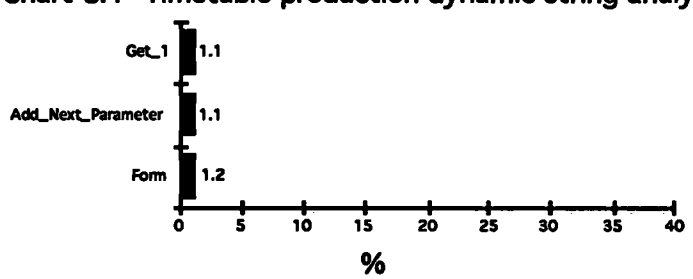

Chart 8.2 Timetable production logic knowledge base analysis.

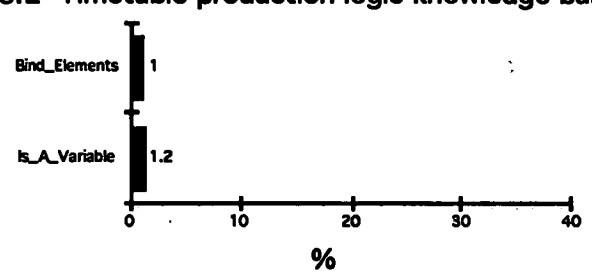

Chart 8.3 Timetable production allocate room knowledge source analysis.

In each of the above cases, the individual operations not shown recorded less than 1% of CPU time.

8.7 Summary

This chapter first shows that abstract knowledge type instances can be integrated into knowledge source processes, as entries on a blackboard or as independent knowledge-based instances that can be interrogated from other components. In addition, the chapter describes an experiment which illustrates these strategies by integrating multiple, diverse and co-operating abstract knowledge types in an automated university timetabling application. The timetable production system first uses an instance of the rule abstract knowledge type to transform the initial input of degree module codes into blackboard entries. These entries then trigger further rule abstract knowledge type instances which produce blackboard entries that represent module requirements, module commonality and module activities such as lectures, tutorials and practicals. At the top level of the blackboard logic abstract knowledge type instances are used to allocate timetable periods and rooms. Frame abstract knowledge type instances are used to represent each period on the timetable and, as independent instantiations, to record the staff and module details. Once the timetable has been created the user may move or swap modules from one period to another through interaction with a graphical interface.

The abstract knowledge types are technically easy to integrate using generic instantiation, but require varying degrees of complexity to use. The use of a rule abstract knowledge type instance is easy, requiring a single call to the inferencing process. Using a frame abstract knowledge type instance is almost as easy, but in this case there are a number of operations that can be selected. However, multiple logic abstract knowledge instances are the most difficult to use since an appropriate query has to be constructed for each instance and the interaction between the multiple instances has to be controlled.

The results of the experiment show that the Dynamic String consumes 72.9% of CPU time. Logic KB and Allocate Room knowledge source are the next largest consumers of processor time, taking 6.1% and 4.5% respectively. All other modules consume 3.3% or less.

Chapter 9

Discussion and Conclusions

9.1 Introduction

Naval researchers, in particular Miles [106], are currently testing a sea-going Data Fusion Technical Demonstrator in order to establish the feasibility of using knowledge-based techniques in future command and control systems. The research work leading to the implementation of the trial concluded that rule-based techniques can be used to solve the data fusion problem associated with the initial process in the command and control system. However, the timing overheads associated with sophisticated knowledge-based development environments, often used to develop knowledge-based solutions, prevented their use in the trial implementation. Consequently, Miles developed a research model to show that knowledge-based data fusion can be implemented directly in Ada, the preferred language in the United Kingdom Ministry of Defence for the implementation of real-time embedded systems. Unfortunately, Miles did not have access to any knowledge-based components written in Ada that could be incorporated directly into his solution. A library of such components would have made his work much easier and would have allowed him to concentrate on the data fusion issues rather than on building the primitive components needed to solve the problem. It was the lack of such components which stimulated this research.

Chapter 1 of this thesis identified a number of potential issues faced by the

designers of future real-time embedded systems that intend to use knowledge-based techniques where conventional algorithmic techniques have proved inappropriate. First, they are faced with the problem of choosing the most appropriate knowledge representation paradigm to model the diverse knowledge domains being encountered knowing that there is no one paradigm that can cope with all knowledge domains; second, having modelled the diverse knowledge domains, they have to decide how the diverse set of knowledge-based components should be controlled; third, they have to choose a programming language in which to implement the knowledge-based components knowing that the most popular languages used for implementing knowledge-based solutions, LISP and PROLOG, are not usually associated with applications that require high reliability and integrity, and need to be maintained over a long period of in-service use. Consequently, Chapter 1 identified three essential requirements needed to simplify the design, implementation and maintenance of future complex embedded knowledge-based solutions; these are to:

- Provide a library of independent software components, to support a variety of knowledge representation paradigms, that can be used to model the diverse expert domains being encountered by the designers of future complex real-time systems. The components should be easy to integrate into different applications, and provide a means of prototyping knowledge-based solutions directly in environments which are dominated by conventional components that have been implemented using procedural languages and software engineering principles.

- Provide the user of the library with a means of creating multiple independent instances of the knowledge-based components, and the means of controlling the instances in order that designers may model problems requiring consultation between multiple co-operating experts.

- Provide a means of integrating and controlling the anticipated complexity of the assembled diverse and possibly multiple knowledge-based components.

Consequently, this thesis set out to show that a variety of independent, generally applicable knowledge representation paradigms can be implemented using the real-time programming language Ada and integrated using a generally applicable control architecture.

Chapter 2 described three of the most commonly used knowledge representation paradigms; logic, rules and frames. Although Ada does not support any knowledge-based primitives, there are aspects of Ada which can be used to implement the proposed knowledge-based components. In particular, the generic package can be used to develop generally applicable abstract knowledge types, while Ada tasks can be used to control the interaction of co-operating components. In addition, the rigour associated with strong typing can be applied, providing the potential for increased reliability and integrity of knowledge-based components.

The case for using the blackboard architecture as the control component was established in Chapter 3. Although hybrid knowledge-based development systems exist, they are not considered appropriate as delivery vehicles in an embedded real-time application; the main problems are their use of LISP as the development language, when Ada is mandated or preferred, together with their inherent slow operating speeds and poor maintenance potential. However, the blackboard architecture was shown to be a generally applicable problem solving structure that can be used to co-ordinate multiple independent processes, where the processes may use any knowledge representation paradigm. In addition, the blackboard can be coded directly in Ada in the same way as the knowledge representation paradigms or

any other software component, providing a uniform implementation approach.

9.2 Abstract Knowledge Type Implementation

Chapters 4 to 7 described experiments to confirm the hypothesis that diverse knowledge representation paradigms can be implemented in Ada in such a way that multiple instances of the abstractions can be easily replicated and used with little effort from designers of future real-time systems. In addition, a means of controlling multiple abstract knowledge type instances, co-operating to solve a common problem, was also discussed. Note that all the abstract knowledge type implementations in this prototype have the minimal functionality required to meet the needs of this research. However, limitations and possible enhancements are discussed in the following sections. These experiments show that the development and use of the abstract knowledge type components is no different to the way conventional abstract data types are developed and used. All experiments were carried out on a VAX 4000 Model 100 running VMS V5.5-2 using VAX Ada V2.3-3. In addition, a version of the experiment was also developed for the Macintosh SE/30 in order to provide an interactive window user environment.

All the code is implemented in Ada and the computer science techniques used in the solutions are within the capabilities of a first degree computer science graduate. Moreover, no low-level code was needed, no unstructured constructs were used and no non-standard features were included. Consequently, the abstract knowledge types are portable.

Each of the abstract knowledge types were tested using the DEC Dynamic Analysis Tool. The specific abstract knowledge type implementation issues and the results of the analysis are discussed in the following paragraphs.

9.2.1 Logic Abstract Knowledge Type

The main concern when designing the logic abstract knowledge type data structures was the need to provide a uniform pointer type across the knowledge base. This was achieved by using a discriminated record to define the different knowledge base node structures thus permitting instances of a single access type to point at variants of the discriminated structure.

The second major concern was the need to control multiple co-operating instances of the logic abstract knowledge type. The Ada task selective wait construct proved to be an excellent solution to this particular problem since a task automatically waits at a select point for an entry request. A user component triggers a logic query by calling the entry of a Solver task. The Solver task finds a solution, transfers it to a Control task and requests whether Any_More solutions are required. The Control task accepts the solution and makes it available to the user component. The user component signifies satisfaction with the solution by responding No_More or requests an alternative solution with Get_More. A Get_More response results in the Solver task discarding the current solution and resuming the search for the next solution.

The potential complexity of the logic knowledge base necessitated the use of recursive subprograms in both the build and inference processes. In particular, recursion was used to form the nested PROLOG structures and to traverse the structures in order to display test results. In addition, the processes for solving goals and unification were also implemented recursively since both involved a traverse of the knowledge base structure. Moreover, the automatic use of the system stack by recursive subprogram calls was a convenient way of storing the goal/match levels discussed in 4.3.2; each goal/match record, and its associated

instance variable and structure pointers, is held in a separate instance of the Solve_Goals subprogram on the system stack.

Instances of the logic abstract knowledge type proved to be an excellent way of implementing the resource allocation processes. In particular, the period and room knowledge sources used logic to deduce the most appropriate resource allocation. The logic control feature was used successfully to achieve co-operation between the different degree timetables when selecting a common module period.

The main limitation of the logic abstract knowledge type is the lack of a list data structure normally found in PROLOG implementations. For example, a list could have been used to record the detail of room usage in the room knowledge base rather than just recording the fact that a room is occupied.

9.2.1.1 Logic Knowledge Base Dynamic Analysis

Table 4.1 shows that the majority of processor time, 85.9%, was spent manipulating Dynamic Strings [16]. In particular, two subprograms consumed most of the time. Chart 4.1 shows that the function New_Structure, which returns a Dynamic String of the required size, consumed 42.0% , and procedure Free, which collects the garbage from discarded strings, consumed 35.2% of the CPU time. The internal analysis of the function New_Structure, which comprises 35 lines, identified the following code fragments as consuming the majority of the CPU time recorded against the function, ail other constructs within New_Structure consumed 0.3% or less. The code fragments show that more time was spent checking the equality of dynamic string sizes than allocating memory to new dynamic strings.

```
if HEADER_INDEX.THE_SIZE = THE_SIZE then        17.4%
while HEADER_INDEX /= null                       8.1%
end loop                                         6.1%
return new SUBSTRING(1..THE_SIZE)                6.1%
HEADER_INDEX := HEADER_INDEX.NEXT                2.0%
```

The internal analysis of the procedure Free identified the following code fragments consuming the majority of CPU time, all other constructs within Free consumed 0.3% or less. Again the equality comparison consumed the most time.

```
if THE_STRUCTURE'LENGTH = HEADER_INDEX.THE_SIZE then        12.8%
elsif THE_STRUCTURE'LENGTH < HEADER_INDEX.THE_SIZE then      9.9%
HEADER_INDEX := HEADER_INDEX.NEXT                            5.3%
while HEADER_INDEX /= null                                   3.5%
PREVIOUS_HEADER := HEADER_INDEX                              2.0%
```

All cases involved variables of the type definitions shown below,

```
type STRUCTURE is access SUBSTRING;
type STRING is
record
   THE_LENGTH      : NATURAL := 0;
   THE_ITEMS       : STRUCTURE;
end record;
type NODE;
type NODE_POINTER is access NODE;
type NODE is
record
   THE_STRUCTURE   : STRUCTURE;
   NEXT            : NODE_POINTER;
end record;

type HEADER;
type HEADER_POINTER is access HEADER;
type HEADER is
record
   THE_SIZE            : NATURAL;
   THE_STRUCTURES      : NODE_POINTER;
   NEXT                : HEADER_POINTER;
end record;
```

where Substring is a generic formal parameter instantiated with Standard.String, and the Node and Header definitions are used to instantiate Free List.

In the knowledge base module, Chart 4.2 shows the procedure Form, which constructs the head of a clause, consumed the majority of the CPU time. Form, which has 20 lines of code, is a simple procedure which involves the dynamic allocation of memory for the Head_Node given in Fig 4.1, and consumed 5.6% of CPU time; this figure is low compared with the Dynamic String operations. The internal analysis of Form shows all the time has been recorded against a single allocator statement.

HEAD := new KB_NODE_RECORD(KIND => HEAD_NODE) 5.6%

The result showing the heavy use of the Dynamic String operations is to be expected since all nodes in the logic knowledge base include a dynamic string. However, the difference between the Dynamic String usage and the time spent in other operations used to build the knowledge base structure is significant.

9.2.1.2 Logic Inference Engine Dynamic Analysis

In the inference process, Table 4.2 shows the majority of time is again consumed by the Dynamic String; 71.0% in this case. The Inference Engine consumed only 19.7% of CPU time with Chart 4.7 showing the majority being spent in the procedure Bind_Elements which is used to bind the clause instances. However, this is small at 3.9% compared with the figures associated with the Dynamic String, where the single procedure, New_Structure, consumed 57.4% of CPU time.

The internal analysis of the procedure Bind_Elements, 113 lines, shows that one statement has the highest time consumption, all other constructs within Bind_Elements consumed 0.4% or less.

BOUND_ELEMENT_PTR := new BIND_RECORD 1.3%

9.2.2 Rule Abstract Knowledge Type

The technique of defining a discriminated record and associated access type used in the logic abstract knowledge type was also used to construct the rule base nodes. However, the rule abstract knowledge type data structures were much easier to implement than those in the logic abstract knowledge type since the syntax of a rule is much simpler than that of a logic clause. Furthermore, the rule abstract knowledge type fact base is implemented as a binary tree of pointers to blackboard entries, which is established by instantiating a generic abstract data type. Consequently, implementation of the fact base required little extra effort.

The linear structure chosen for the rule knowledge base meant that an iterative search could be made during the inference process, which was much simpler to comprehend than the recursive processes involved in the matching and unification of the logic goals. Moreover, unlike the logic abstract knowledge type where the preparation of a query involved constructing complicated strings, the rule inference process is triggered by an event occurring on the blackboard.

Instances of the rule abstract knowledge type are used to represent knowledge of the structure, commonality and activity requirement of degree modules. The rules transform primitive inputs of degree module codes into multiple activities which trigger the period allocation process.

As indicated earlier, this implementation provides the minimum functionality to support this research. Consequently, there are many variations that can be implemented to improve efficiency and provide reasoning in domains where the facts may be uncertain. One advantage of adopting the abstract knowledge type approach to the implementation of knowledge-based components is that such

variations can be encapsulated as different library versions in a similar way to a library of search algorithms. In this way, developers would be able to choose between simple and sophisticated implementations to match the application.

9.2.2.1 Rule Base Dynamic Analysis

Table 5.1 shows the Rule Base was also dominated by Dynamic String, in this case consuming 64.3% of CPU time. In particular, Chart 5.1 shows Free consumed 24.5% and New_Structure 20.9%. The reduction from 85.9% to 64.3 % can be explained by the fact that far fewer strings were generated in the rule base than in the logic knowledge base. Chart 5.2 shows the I/O operation Get from System Types consumed 16.6% of CPU time. However, this did not appear in the logic abstract knowledge type, where the Get operation in Knowledge Base consumed only 1.5%. On examination of the two Get operations, two major differences became apparent: first, the logic abstraction reads a whole clause in a single Get operation which is terminated by detecting a '.'; second, the Get operation in Rule Base reads a single token and, in some cases, uses Text_IO.End_Of_Line to detect the token terminator. Consequently, the approach used for input in the logic abstract knowledge type is more efficient than that used in the rule abstract knowledge type.

The internal analysis of System Types Get shows the following code fragments consumed the majority of the CPU time. All other constructs consumed 0.2% or less.

TEXT_IO.GET	10.8%
TEXT_IO.END_OF_LINE	4.8%
DYNAMIC_STRING.APPEND	0.7%

Chart 5.3 shows the main time consumer in the rule base component is procedure

Build, comprising 98 lines, which consumed 5.0% of CPU time. Furthermore, the internal analysis of Build shows that the following code fragments consumed most time. All other constructs within Build consumed 0.1% or less.

TEXT_IO.END_OF_FILE	1.0%
TEXT_IO.CLOSE	0.7%

9.2.2.2 Rule Inference Engine Dynamic Analysis

Table 5.2 shows inferencing is again dominated by Dynamic String which used 47.8% of CPU time. In particular, Is_Equal consumed 18.1%, Free 11.4% and New_Structure 10.5%. Furthermore, since the test builds a rule base prior to inferencing, System Types Get recorded 12.4% and Rule Base Build 4.8%. During the inference process, Chart 5.8 shows only 8.6% of CPU time is consumed in forming the agenda.

9.2.3 Frame Abstract Knowledge Type

The main concern when designing the frame abstract knowledge type was to provide a flexible and efficient frame/slot/facet structure. This was achieved by using an N-ary tree to represent the frames and nested binary search trees to represent the slots and facets. In addition a binary search tree overlays the frame tree in order to provide an efficient frame search mechanism, since the frames are connected by inheritance relationships rather than in an ordered sequence.

A similar discriminated record and access type definition to that used in the logic and rule abstract knowledge types was used to define the frame base nodes. However, unlike the logic and rule abstract knowledge types where the node structures are application independent, the frame facet was made generic so that the

type of the stored knowledge could be determined by the user component. This caused problems when trying to instantiate a generic instance of the frame abstract knowledge type since each facet tree was an element of a slot tree, which in turn was an element of a frame tree. This problem is discussed in 9.2.5.2.

The majority of the frame base operations were performed by using combinations of the operations provided by the underlying tree instances. Consequently, only a little extra effort was required to develop the frame abstract knowledge type operations. However, as with the logic and rule abstract knowledge types, the frame abstract knowledge type has restricted functionality. For example, demons could be added to react to changes made to the frame contents. This suggests different library versions could be produced as was proposed for the rule abstract knowledge type.

Instances of the frame abstract knowledge type were used successfully as independent components to represent the university staff and module details. Moreover, frame abstract knowledge type instances were used to represent the timetable period structure on the top level of the blackboard and, although not implemented, any abstract knowledge type instance could be defined as an element of a frame instance itself.

9.2.3.1 Frame Base Dynamic Analysis

Chart 6.1 shows that the Frame Base Build operation dominated the frame test process, unlike logic and rule where Dynamic String consumed most of the CPU time. In the frame Build operation the internal analysis shows that two statements consumed the majority of CPU time:

new FACET_RECORD_TYPE	45.6%
TEXT_IO.END_OF_FILE	4.8%

Expanding the allocator, given below, shows the use of Dynamic String Substring_Of to convert Name into a enumerated value of Slot_Type, which is then used as the discriminant for the record allocator.

```
CURRENT_FACET_PTR := new FACET_RECORD_TYPE.
                          ( SLOT_KIND =>
                            SLOT_TYPE'VALUE
                            ( SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              SUBSTRING_OF
                              (THE_STRING =>  CURRENT_SLOT_PTR.
                                              NAME));
```

Although Frame Base Build dominated this test, Dynamic String still consumed a significant amount of time, recording 17.7% CPU usage. The reduced influence of Dynamic String is due the fact that fewer strings were generated when compared with the logic abstract knowledge type. However, when compared with the rule abstract knowledge type, where more dynamic strings were created, the reduction in influence can be explained by the increased complexity of the facet allocator compared with the rule node allocator.

9.2.4 Blackboard Abstract Knowledge Type

The main concern when designing the generic blackboard data structure was to provide a flexible architecture that was easy to construct, but at the same time provide sufficient representational power to satisfy complex application requirements. This was achieved by using unconstrained arrays to represent the vertical and horizontal dimensions of the blackboard. In each case the arrays are constrained by the user component in the definition of an application dependent enumerated type. In addition, each blackboard level array location can be instantiated to contain any abstract data type or abstract knowledge type that is available. This strategy proved to be very effective and easy to use. Moreover,

although the blackboard structure appears complex, only three operations are needed in order to achieve the resuit.

No major problems were anticipated with the blackboard abstract knowledge type test. Table 7.1 shows the percentages of CPU time distributed almost evenly between the blackboard construction and initialisation operations.

Within Initialise Blackboard, Chart 7.1 shows the major consumer of CPU time was the procedure Initiaiise_Level, where the horizontal dimensions passed as actual parameters, are used to constrain the level's upper and lower bounds. The internal analysis of Initialise_Level, comprising 38 lines of code, showed the allocator

```
NEW_ITEM := new     TEST_BB_TYPES_PACKAGE.
                    BLACKBOARD_ITEM_STRUCTURE_RECORD      22.8%
                    (ITEM_STRUCTURE_TYPE);
```

consumed 22.8% of the CPU time; this allocator sets the content of each component of a blackboard level to the required item structure. All other constructs in Initialise_Level consumed 0.8% or less. Within Test_Biackboard, Chart 7.2 shows that the operation Construct_Blackboard_Level, comprising 16 lines of code, consumed 16.5% of CPU time; Construct_Blackboard_Level is responsible for allocating memory to the horizontal array of the specified level. The internal analysis of Construct_Blackboard_Level shows that the allocator

```
LEVEL_PTR := new    BLACKBOARD_HORIZONTAL_ARRAY_TYPE
                    (FROM .. TO);                         14.2%
```

consumed 14.2 % of CPU time. All other constructs in Test_Biackboard consumed 0.8% or less.

9.2.5 Ada Implementation Issues

9.5.2.1 Type Definitions

Internal type definitions are used to model the appropriate abstract knowledge type data structures using unconstrained arrays and discriminant records to provide the flexibility needed to construct efficient implementations. The unconstrained array is particularly useful in cases where the size of the array instances vary, but each instance must be of the same type. For example, this technique is used to define the storage needed for clause variables in the Instance_Variables_Record located in the logic abstract knowledge type, so that allocated storage matches exactly the number of variables in a particular clause. In addition, an unconstrained array is an ideal way to define the blackboard, where the levels are constrained to model the application requirement exactly, although the size of each level is different.

A discriminant record is used to define variants of the same record type. This is useful in the construction of the complex linked structures needed to implement the abstract knowledge types. This technique is used in all the abstract knowledge types to define the component parts of the complex structures. This is necessary since the single access type definition needed to link the nodes must refer to a single type, although the nodes themselves have different internal structures.

Ada strong typing and the requirement to define before use is often cited as a disadvantage when prototyping solutions. Although the requirement to define all types before use takes time, this is often not difficult, and the declaration process also helps the implementor focus on the structure of the implementation to form a very clear mental image of the complex structures being manipulated.

9.2.5.2 Dynamic String

The analysis shows that the operations from Dynamic String consume a significant amount of CPU time in both the logic and rule abstract knowledge types, with less influence in the frame abstract knowledge type. In the logic component the Dynamic String operations consume 71.0% of CPU time whiie in the rule component the Dynamic String consumed only 47.8%. The difference in percentages can be explained by the fact that an average sized logic clause is constructed from more dynamic strings than is a rule. However, the analysis shows the Dynamic String to be a critical component when considering real-time operation. This problem could be addressed by providing a translator that produced string tokens and a symbol table. The tokens would be stored in the knowiedge bases and used in the inference processes to reduce string manipulation times. However, the literal strings would still be needed: in the logic abstract knowledge type the strings would be needed to report results and formulate queries; in the rule abstract knowledge type the strings would be needed to support an explanation facility and to report rule actions; in the frame abstract knowledge type tokens could be used to reduce storage requirements, but the dynamic strings would be needed for reference.

9.2.5.3 Generic Packages

Having established the form of the data structures for each abstract knowledge type, implementation of the Ada package specifications was straightforward. Generality and multiplicity is achieved by using the Ada generic construct. Where appropriate, generic parameters were defined to enable an application to tailor the abstraction to meet application requirements and to permit the instantiation of multiple independent copies of a component. For example, the Inference Engine in the logic abstract knowledge type requires just one generic parameter; the generic

name of the knowledge base to be supplied by the application.

However, where an abstract knowledge type encapsulates other generic components then the generic definition is more complex. This situation arises when an embedded generic component needs to be instantiated with one or more of the generic parameters defined for the encompassing abstract knowledge type.

In the rule abstract knowledge type the fact base is supported by a generic tree; this requires actual generic parameters for the data stored in the tree, together with subprogram generic parameters to permit the tree manipulation algorithms to process the generic data. In this case instantiation was straightforward, although more actual generic parameters were needed when compared with a logic instantiation.

The frame abstract knowledge type specification is further complicated by the need for three levels of generic instantiation in order to provide operations that traverse the complex structure. The resulting generic instantiation and call structure for the frame abstract knowledge type is illustrated in Fig 9.1. First, an instance of the generic Tree is instantiated with the generic actual procedure parameter Put to support the Facet structure; the resulting Facet Tree operation, Put1, is then used to define the actual subprogram parameter, Put2, for the Slot package. Put2 is then used to instantiate a second instance of the generic Tree to support the Slot Tree structure; the resulting Slot Tree operation, Put3, is then used to define the actual subprogram parameter, Put4, for the Frame package. Finally, Put4 is used to instantiate a third instance of the generic Tree to support the frame Search Tree structure, which results in an instance of the Tree operation, Put5. Finally, a frame base traverse routine, Put 6, is implemented which calls either Put4 or Put5.

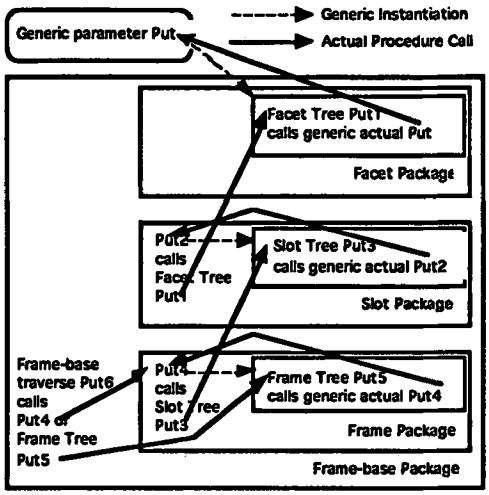

Fig 9.1 Frame Base instantiation and call structure.

To traverse the structure down the frame hierarchy, Put 6 calls Put4, which results in the call chain Put4 -> Put3 -> Put2 -> Put1 -> Put, for each frame in the hierarchy. Alternatively, if Frame Tree is instantiated with Put3 an ordered traverse is accomplished by calling Put 5, which results in the call chain Put5 -> Put3 -> Put2 -> Put1 -> Put for each frame in the frame search tree.

9.2.5.4 Tasks

Chapter 4 shows how Ada tasks can be used to control the dialogue between multiple logic abstract knowledge type instances; the dialogues are needed in order to solve complex issues involving a number of experts. The implementation of such a control strategy is stralghtforward, requiring only two tasks: the first task performs the inference algorithm and can be suspended at any point a solution is found; the second task acts as an interface between the inference task and the user component. Generic instantiation of the logic abstract knowledge type creates an independent instance which can be in any state relative to other instances. Consequentiy, each logic instance can be solving separate problems, but able to co-operate and share knowledge to solve an overall problem. This technique is

demonstrated in the timetable experiment where each timetable is organised by separate logic instances. However, where modules share a common syllabus the logic instances co-operate to agree a timetable period allocation.

Although this technique was implemented in the logic abstract knowledge type, the concept is generally applicable to any component which needs to be suspended at a possible solution point in order that the current solution can be agreed with one or more other abstract knowledge type instances. For example, this concept can be applied in a rule abstract knowledge type so that at any point, forward chaining can be suspended, another abstract knowledge type instance consulted, and the forward chaining process re-activated.

The Ada tasking facility is excellent for controlling the interaction between co-operating abstract knowledge type instances. In addition, because tasks are concurrent program units, parallel consultation is possibie, but not implemented.

9.2.5.5 Compilation and Debugging

Once an Ada specification is complete it can be compiled separately from its body. This approach is recommended since separate compilation establishes confidence in the specification and also permits the separately compiled body to be re-implemented without the need to re-compile the specification. Compilation of strongly typed procedural languages is another characteristic often cited as a disadvantage and a reason not to use the procedural paradigm in the development of knowledge-based solutions. This is a problem on machines such as the Macintosh SE/30 using Meridian Ada, where the final experimental code took 20 minutes to compile. However, on the VAX , using DEC Ada, compilation time was insignificant. Moreover, the Ada compiler diagnostics were easy to use, resulting in the quick

correction of syntactical errors. In addition, the strong type checking established confidence that the code would perform as intended and any run time issues were resolved quickly using a symbolic debugger.

The use of Ada brings discipline to the implementation of the abstract knowledge types. The extra time needed to set up the specifications is seen as an advantage rather than a disadvantage since the process helps the implementor focus onto the complexity of the solution. In addition, the compilation times were insignificant and debugging easy and quick.

9.2.5.6 Libraries

Ada compilation environments provide comprehensive built-in program unit library and compilation management routines. Consequently, no special effort is required by an implementor to create libraries of abstract knowledge types, the abstract knowledge types are handled in exactly the same way as any other Ada component.

9.2.5.7 Language Restrictions

The constructs provided in Ada 83 have been found very suitable for constructing a set of prototype abstract knowledge types. However, problems are anticipated should Ada 83 be the only choice for future implementations. In particular Ada 83 lacks an effective and simple means of extending the abstract knowledge type definitions in a similar way to that available in object-oriented languages, such as C++ and Eiffel, which provide classes, subclasses and inheritance.

The superior extension capabilities of the object-oriented languages would be useful

in defining the variations on the abstract knowledge types proposed earlier. This would be achieved by constructing a primitive abstraction followed by a number of more sophisticated versions derived through the definition of subclasses; the subclasses inherit the properties of the primitive abstraction and add the required complexity. Moreover, in the current implementation of the knowledge sources, extending the range of degrees would be time consuming and require changes to the existing code. In an object-oriented solution each degree would be represented by an instance of a class of degrees having a set of operations that could be applied to any degree instance. Extending the range of degrees would then be a matter of instantiating a new degree object and injecting it into the system. Fortunately, Ada 9X provides these object-oriented facilities. Finally, although this research has been driven by the Ada mandate the abstract knowledge types could be developed in languages such as Eiffel and C++ for use in other application domains.

9.2.6 Research Related to Abstract Knowledge Types

Several papers have been published which describe experiments that have used Ada 83 to implement rule-based tools and components. Jaworski [83] and Hall [64] describe an expert system tool to assist in the production of Ada rule-based solutions. However, their approach is to develop the solution in a LISP environment and transform the result into Ada in a similar way to the approach described by Hintz [76]. The disadvantage of this approach is that implementors need to be proficient in both the prototyping and implementation languages. Moreover, some maintenance will have to be done in the prototyping environment when 'pure' Ada is mandated or preferred. The approach proposed in this thesis is to develop a library of pre-defined abstract knowledge type components and use instances of the components to build prototypes in the application environment. This enables implementors to treat the abstract knowledge types in exactly the same way as any

other software component and only the mandated or preferred language is involved in development and maintenance.

Adkins [2] and DeFeyter [47] investigated the use of Ada tasks for implementing the rule-based paradigm. Adkins represents each rule by a task and concludes that the software "required a great deal of experimentation to find even a single working configuration"; the main problem was controlling the task start and stop conditions. DeFeyter however, uses tasks to represent message passing expert objects comprising several internal rules. This approach has similarities to the idea of abstract knowledge types, but is limited to rule representation. However, the concept could be extended so that the knowledge represented inside each object can be any paradigm. In fact, the task objects could use the abstract knowledge types as the basis for their internal knowledge representation. Consequently, the message passing expert object approach can be viewed as an alternative way of integrating the independent abstract knowledge type instances.

Hirshfield [77] Labhart [95] and Wright [150] describe the implementation of rule-based inference engines for use in real-time embedded systems. The techniques used in these experiments can form the basis for variations on a class of abstract knowledge types, where the alternatives exhibit different operational characteristics, giving developers a choice of inferencing or implementation strategy. This approach is analogous to the idea of mathematical function libraries where similar functions are provided, but which exhibit slightly different implementation characteristics giving different memory/time profiles. The need for a variety of approaches can be deduced from the paper by Wiiber [145], which is written in the context of avionics systems, where he states that "The architectures selected to solve the real-time embedded problem requires only basic knowledge-based techniques"; however, other applications, for example command

and control, will need more sophisticated techniques so a variety of implementations will be needed.

The work by Wallnau [144] comes nearest to the ideas proposed in this thesis since Wallnau also suggests the implementation of knowledge-based components as abstract data types; Wallnau describes a rule-based inference system and a structured inheritance network. In addition, Wallnau uses special purpose knowledge base description languages to instantiate the component configuration required by an application. The translators for these languages generate the Ada programs which create the knowledge base instances using calls to the knowledge-based components. The knowledge-based description languages are being used to construct a knowledge-based testing assistant and a domain specific software reuse library and Librarian. In a real-time embedded application, the extra complexity generated by the description language approach is not necessary since the abstract knowledge types can be instantiated directly into the application code. Furthermore, it is important that the abstract knowledge types should be used in exactly the same way as any other software component. However, this research supports the use of abstract data type techniques for building the knowledge-based components and extends the idea by identifying the need to include a generally applicable control component in the abstract knowledge type library. In addition, Kimble [89] states that "Capturing the many existing good AI algorithms in a common repository would be a plus for the new systems to be implemented in Ada".

Several papers have also been written which describe the Ada implementation of logic components. For example, Ice [79] , Dobbins [49] and Kilpelainen [88] describe sequential implementations of PROLOG similar to that presented in Chapter 4, whereas Bobbie [14, 15] uses Ada tasks to implement a parallel PROLOG where each predicate is satisfied concurrently. Furthermore, Harding [65] uses tasks to

separate the inferencing and search activities. Other logics have been implemented, in particular, Baker [7] describes a non-Horn clause logic interpreter which uses both backward and forward chaining. In addition, Burback [24] describes a first-order predicate logic interpreter. These are all examples of the diversity of abstract knowledge types that could be supplied in a library of knowledge-based components.

Finally, Stockman [138] describes an Ada-based blackboard system designed to provide Ada applications with blackboard functionality. Furthermore, Stockman's proposal is that the blackboard requirement be defined in a LISP based language and translated into Ada. However, this thesis proposes the blackboard be implemented in the same way as the other abstract knowledge types so that all the knowledge-based components may be used in the same way as conventional software components.

This research has shown that building an abstract knowledge type in Ada 83 is no different from building an abstract data type. Consequently, the abstract knowledge types can be engineered to the same standard as other software components by applying the same software engineering principles. Moreover, there is no evidence to suggest that other abstract knowledge types cannot be constructed in the same way and added to the library of components.

9.3 Abstract Knowledge Type Integration

Integration of the blackboard is achieved by a simple generic instantiation which requires the previous Ada specification of the application blackboard entries, and enumerated type definitions giving the values for the application levels and horizontal divisions. No major problems were encountered when using the

blackboard instance. Although only one blackboard instance has been used in this experiment, the generic implementation permits multiple instantiation, if required, by an application. For example, it would be easy to instantiate two blackboards, one to represent control knowledge and the other to represent domain knowledge, as described in Chapter 3.

Three logic abstract knowledge type instances, ESE_Period_Inference_Package, IT_Period_Inference_Package and Room_Inference_Package, have been integrated using the simple instantiation described in Chapter 5. However, although querying a logic instance is simple, the preparation of the query is more complicated; the appropriate goal has to be constructed prior to applying the query. Furthermore, using the logic abstract knowledge type control mechanism to communicate with multiple instances to come to an agreed solution is straightforward. No problems were encountered with the Ada tasking model.

Four rule abstract knowledge type instances were integrated in this experiment; one in each of the Module_Structure_KS, Module_Requirements_KS, Common_Modules_KS and Module_Activity_KS. The instantiation process is slightly complicated by the need to provide the detail for the underlying Tree ADT. However, a rule base query is simple to implement, but the user component is required to process the rule agenda.

Two independent frame abstract knowledge type instances were created; Staff and Module, which are used by various knowledge sources to check staff and module details. Like the rule abstract knowledge type, the instantiation of a frame instance is complicated by the need to provide the detail for the underlying Tree ADT. In addition, application dependent frame operations have to be defined by the user component to manipulate the frame instances. Finally, a frame instance is included

in each of the timetable period blackboard entries, a further illustration of the multiple use of a knowledge representation paradigm in a single application.

9.3.1 The Integration Experiment

The results given in Table 8.4, which show that Dynamic String consumed 72.9% of CPU time, confirm the findings from the abstract knowledge type experiments. Furthermore, the internal analysis given in Chart 8.1 shows the function New_Structure exhibiting similar timing characteristics to those found in the earlier experiments; these are:

if HEADER_INDEX.THE_SIZE = THE_SIZE then	14.3%
HEADER_INDEX := HEADER_INDEX.NEXT	8.2%
return new SUBSTRING(1..THE_SIZE)	6.7%
end loop	5.3%
while HEADER_INDEX /= null	1.8%
PREVIOUS_HEADER := HEADER_INDEX	1.1%

The most prominent abstract knowledge type is Logic KB, which consumed 6.1%; this component contains the knowledge base operations for all instances of Logic Inference Engine. However, Chart 8.2 shows the procedure Form, which constructs a knowledge base clause, contributed the maximum CPU time for the component, but this is only 1.2%. In addition, Table 8.4 shows the inference operation from Allocate_Room_KS consumed 4.5%, but the contribution from individual operations is 1.2% or less.

9.3.2 Research Related to Abstract Knowledge Type Integration

No record of any research could be found in the literature which specifically focuses on the integration of knowledge-based components in Ada. As indicated earlier,

Wallnau's work employs two paradigms, but uses the components to support a software reusability library framework rather than exploring the provision of a library of diverse knowledge representation paradigms; the abstract knowledge types proposed in this thesis would become entries in Wallnau's reuse library!

9.4 Research Contribution

This thesis presents the following as contributions to the field of knowledge regarding the implementation and integration of multiple and diverse knowledge representation paradigms:

- The use of abstract knowledge types, which combine the characteristics of abstract data types and knowledge-based techniques, is proposed as the way of implementing diverse knowledge representation paradigms for use in complex real-time embedded applications.

- Three generally applicable abstract knowledge types capable of multiple independent instantiation have been built and tested.

- A blackboard architecture has been implemented in the same way as the abstract knowledge types and used to integrate and co-ordinate multiple instances of the abstract knowledge types.

- Ada tasks have been used in the logic abstract knowledge type to provide an independent control interface for each logic instance so that multiple instances can be co-ordinated to solve a common problem.

- The abstract knowledge types have been used in a university timetabling experiment which confirms the feasibility of the proposal.

9.5 Conclusions

By analysing the discussion presented in the previous chapters it is possible to draw the following conclusions from this research:

- A library of independent software components, that supports a variety of knowledge representation paradigms, is needed so that the components can be used to model the diverse expert domains being encountered by the designers of future real-time embedded systems.
- The user of the library will need to be able to create and control multiple independent instances of the knowledge-based components in order to model problems requiring consultation between co-operating experts.
- A generally applicable problem solving component is required to provide a means of integrating and controlling the anticipated complexity of the assembled diverse and possibly multiple knowledge-based components.
- A range of knowledge representation paradigms and associated inference algorithms can be implemented in a similar way to conventional abstract data types where the data structures and operations of the abstract data type are replaced by the knowledge base and inference operations of the knowledge representation paradigm. These knowledge-based components have been called abstract knowledge types.
- The marriage of knowledge-based techniques with the concept of an abstract data type to form the abstract knowledge type coalesces the advantages of both approaches.
- The most popular languages used for implementing knowledge-based solutions are either LISP or PROLOG. These languages are not suitable for implementing solutions which require the high reliability, integrity and maintainability associated with real-time embedded systems.

- For many solutions, particularly in real-time embedded systems where reliability, integrity and ease of maintenance is required, Ada is either mandated or preferred as the implementation language.
- Using a hybrid knowledge-based development environment containing an Ada code generator is one way of providing embeddable solutions. However, this is not a 'pure' Ada approach and will contravene the Ada mandate. In addition, the resulting code will be difficult to maintain.
- Abstract knowledge types can be implemented directly in Ada. This gives a 'pure' Ada solution which permits the abstract knowledge types to be used in exactly the same way as conventional components.
- The Ada generic package construct is excellent for implementing generally applicable abstract knowledge types that are easily replicated to form independent embeddable components.
- The Ada task provides an easy way of implementing the control of independent, co-operating instances of abstract knowledge types.
- The Dynamic String abstract data type operations used in all the abstract knowledge types consumed significant amounts of CPU time. An alternative form of representation is needed for use in operational components. A transformation process that produces a symbol table of dynamic string tokens could be used to overcome this problem.
- Ada environments provide an inherent library facility for managing the abstract knowledge types.
- Compilation times were not significant and debugging was quick and easy.
- The blackboard architecture is an effective means of integrating and controlling the abstract knowledge type instances.
- The abstract knowledge type instances can be placed in independent knowledge sources, shared between knowledge sources or placed on the blackboard.

- Abstract knowledge type instances were easy to integrate into other components.
- The abstract knowledge type approach will enable designers of future real-time systems intending to use knowledge-based techniques to prototype their solutions in a 'pure' Ada environment without the need to spend significant amounts of time constructing the knowledge-based components.
- The abstract knowledge type approach provides a single uniform implementation strategy. That is, all software components are implemented, stored, instantiated and used in the same way.
- Variations on classes of abstract knowledge types can be implemented using object-oriented techniques to form a selection of components ranging from simple to sophisticated.
- The availability of abstract knowledge type components provided the opportunity to construct a novel solution to the university timetable problem.
- Timetable production evolved through a series of explosive and fusion processes.

9.6 Future Work

Although the data structures and operations of an abstract knowledge type are more complex than those associated with abstract data types, the abstract knowledge types are built and instantiated in the same way as abstract data types. This association of abstract knowledge types with abstract data types suggests that it would be useful to explore the feasibility of formally specifying the abstract knowledge type components.

Three abstract knowledge types were built in order to test this thesis. Further experiments need to be carried out with variations on these components. For example, the addition of a backward chaining inference mechanism to the rule abstract knowledge type and the development of a range of versions, from simple to sophisticated, for each component. In addition, other abstract knowledge types, for example, neural networks should be added to the library.

All the abstract knowledge types constructed in this thesis use dynamic strings, which consume significant amounts of processor time. Although the Booch dynamic string was convenient at the time the experiments were conducted, some effort needs to be expended to develop a component which will transform the strings into a more computationally efficient form. This should be done in such a way that the same component can be used by all abstract knowledge types.

While constructing the abstract knowledge types it was apparent that the data structures were similar, for example logic is similar to the rule abstract knowledge type, and the frame is similar to the blackboard. It would have been useful if there had been a generic component that unified these structures and was capable of instantiation to the required form.

Finally, Ada 9X will be available in 1995. The experiments described in this thesis and the construction of additional abstract knowledge types need to be implemented in this new language to take advantage of the new object-oriented features.

Bibliography

1. Ablett, S.: '*Clustering Round The Blackboard*', 1986, Infomatics, p. 61-63.

2. Adkins, M.M.: '*Flexible Data Structures in Ada*', 1986, in *AIDA:The 2nd Annual Conference Artificial Intelligence and Ada,* George Mason University USA, p. 9-1 to 9-17.

3. Aiello, L.C. and D. Nardi: '*Perspectives in Knowledge Representation*', 1991, p. 29 - 44.

4. Aletan, S.: '*Current and Future Trends in Artificial Intelligence Architectures and Programming Languages*', 1989, in *Tools for Artificial Intelligence Architectures, Languages and Algorithms,* p. 215 - 221.

5. Arlabosse, F., *et al.*: '*Blackboard and Alternative Architectural Design for Two Real-Scale Industrial Applications*', 1989, in *Blackboard Architectures and Applications,* V. Jagganathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 433-454.

6. Bachimont, B.: '*DOTMS: A Dynamic Object-Based Truth Maintenance System to Manage Consistency in a Blackboard*', 1991, in *Expert Systems & Their Applications,* Avignon(France), p. 109-122.

7. Baker, L.: '*Non-Horn Clause Logic Programming in Ada*', 1988, in *AIDA:The 4th Annual Conference Artificial Intelligence and Ada,* George Mason University, p. 16-1 to 16-7.

8. Baker, L.: '*Artificial Intelligence With Ada*', 1989, McGraw-Hill.

9. Baruah, S.K., *et al.*: '*A Blackboard Architecture to Support Generation of Schematics for Design Automation*', 1989, CAD Systems Using AI Techniques, p. 51-58.

10. Baum, L.S.: '*Recent Developments in Blackboard Frameworks*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and S.L. Baum, Academic-Press, p. 303-308.

11. Baum, L.S., R.T. Dodhiawala, and V. Jagannathan: '*The Erasmus System*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 347-370.

12. Bench-Capon, T.J.M.: '*Knowledge Representation*', 1990, Academic-Press.

13. Bisiani, R. and A. Forin: '*Parallelization of Blackboard Architectures and the Agora System*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and L.S. Baum, p. 137-152.

14. Bobbie, P.O.: '*Ada-Prolog: An Ada System for Parallel Interpretation of Prolog Programs*', 1987, in *AIDA:The 3rd Annual Conference on Artificial Intelligence and Ada,* George Mason University, USA, p. 102-123.

15. Bobbie, P.O. and C. Cowden: '*On the Implementation of a Parallel Prolog Interpreter in Ada*', 1988, in *AIDA-88: The 4th Annual Conference on Artificial Intelligence and Ada,* George Mason University, USA, p. 5-1 to 5-23.

16. Booch, G.: '*Software Components With Ada*', 1987, Addison-Wesley.

17. Brachman, R.J. and H.J. Levesque: '*Competence in Knowledge Representation*', 1982, in *AAAI,* p. 189 - 192.

18. Brachman, R.J., V.P. Gilbert, and H.J. Levesque: '*An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of Krypton*', 1985, in *IJCAI,* p. 532 - 539.

19. Brachman, R.J.: '*The Future of Knowledge Representation*', 1990, in *8th Conference on Artificial Intelligence,* p. 1082-1092.

20. Brachman, R.J.: '*What is Knowledge Representation, and Where Is It Going?*', 1992, in *Future Tendencies in Computing Science Control and Applied Mathematics*, p. 189-203.

21. Brander, G.N. and E.J. Bennett: '*Real-time Rule-based Resource Allocation in Naval Command and Control*', 1991, in *IEE Colloquium on "Rule-based Real-time Planning and Control"*, Savoy Place London, p. 1-6.

22. Bratco, I.: '*Prolog Programming For Artificial Intelligence*', 1987, Addison-Wesley.

23. Brauer, D.C., *et al.*: '*Ada and Knowledge-Based Systems: A Prototype Combining the Best of Both Worlds*', 1986, in *Expert Systems in Government*, p. 198 -202.

24. Burback, R.: '*PROVER: A First-order Logic System in Ada*', 1987, in *AIDA:The 3nd Annual Conference Artificial Intelligence and Ada*, George Mason University, USA, p. 166-190.

25. Butler, G. and M. Corbin:'*FLEX: A Fortran Library for Expert System Development*', 1988, Royal Aerospace Establishment, Farnborough.

26. Buttner, K., D. Sriram, and M. Freiling: '*An Object-Oriented Blackboard Architecture for Model-Based Diagnostic Reasoning*', 1989, in *Blackboard Architectures and Applications*, V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 403-431.

27. Cacciabue, P.C., *et al.*: '*A Cognitive Model in a Blackboard Architecture: Synergism of AI and Psychology*', 1992, Reliability Engineering and System Safety, p. 187-197.

28. Carlson, W.:'*Programming Language Ada, Language and Standard Libraries*', 1993, International Organization for Standardization International Electrotechnical Committee Information Technology Programming Languages Their Environment and System Software Interfaces.

29. Chouvet, D., *et al.*: '*CORADA: An Expert System Compiler into Ada*', 1989, in *Ada Europe*, p. 237 - 244.

30. Clocksin, W.F. and C.S. Mellish: '*Programming in Prolog*', 1987, Springer-Verlag.

31. Collard, P.E. and A. Goforth: '*Ada in AI or AI in Ada? On Devloping A Rationale For Integration*', 1988, in *4th Conference on Artificial Intelligence for Space Applications*, p. 411-419.

32. Collard, P. and A. Goforth: '*Knowledge Based Systems and Ada: An Overview of the Issues*', 1988, Ada Letters, p. 72-81.

33. Collins, W.R. and L.P. Slothouber: '*Expert Systems Control in Ada*', 1988, in *AIDA:The 4th Annual Conference Artificial Intelligence and Ada*, George Mason University, p. 17 -1 to 17-11.

34. Corby, O.: '*Blackboard architectures in computer aided engineering*', 1986, Artificial Intelligence, p. 95-98.

35. Corkill, D.D., K.Q. Gallagher, and K.E. Murray: '*GBB:A Generic Blackboard Development System*', 1986, in *5th National Conference on Artificial Intelligence*, USA, p. 1008-1014.

36. Corkill, D.D.: '*Advanced Architectures:Concurrency and Parallelism*', 1989, in *Blackboard Architectures and Applications*, V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press, p. 77-83.

37. Corkill, D.D.: '*Design Alternatives for Parallel and Distributed Blackboard Systems*', 1989, in *Blackboard Architectures and Applications*, V. Jagannathan, R. Dodhiawala, and L.S. Baum, p. 99-136.

38. Corkill, D.D.: '*Embedable Problem-Solving Architectures: A Study of Integrating OPS5 with UMass GBB*', 1991, IEEE Transactions on Knowledge and Data Engineering, p. 18-24.

39. Corkill, D.: '*Blackboard Systems*', 1991, AI Expert, p. 41-47.

40. Craig, I.D.: '*The Ariadne-1 Blackboard System*', 1986, The Computer Journal, p. 235-240.

41. Craig, I.D.:'*The Blackboard Architecture: A Definition and its Impications*', 1987, University of Warwick.

42. Craig, I.D.: '*Blackboard Systems*', 1988, Artificial Intelligence Review, p. 103-118.

43. Craig, I.D.: '*Decentralised Control in a Blackboard System*', 1988, University of Lancaster.

44. Daniel, J.W.H. and J.M. Haugh:'*A Survey of Knowledge Engineering Tools for the Data Fusion Project*', 1987, Admiralty Research Establishment.

45. Davison, S.J. and R. Kraft: '*COGSYS : Real-Time Decision Support For Process Control*', 1990, Proceedings of the Tenth International Conference 'Expert Systems and Their Applications', p. 23-37.

46. Decker, K.S., V.R. Lesser, and R.C. Whitehair: '*Extending a Blackboard Architecture for Approximate Processing*', 1990, The Journal of Real-Time Systems, p. 47-79.

47. DeFeyter, A.R.: '*RTEX: An Industrial Real-Time Expert System Shell*', 1988, in *AIDA:The 4th Annual Conference Artificial Intelligence and Ada,* George mason University, USA, p. 6-1 to 6-22.

48. Diaz-Herrera, J.L.: '*The Ada-AI Interface*', 1987, in *AFIPS,* Chicago, USA, p. 67-72.

49. Dobbins, G.L., V.E. Szarek, and W.H. Webster: '*An Ada/Prolog Database Management System For Ada Source Code*', 1987, in *AIDA:The 3nd Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 23-39.

50. Dodhiawala, R.T.: '*Blackboard Systems in Real-Time Problem Solving*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, p. 181-189.

51. Dodhiawala, R.T., N.S. Sridharan, and C. Pickering: '*A Real-Time Blackboard Architecture*', 1989, in *Blackboard Architectures and Applications*, V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, p. 219-237.

52. Dunkelberger, K.A.: '*Organization for combat: a deliverable military planning aid*', 1989, p. 725 - 734.

53. Elfes, A.: '*A Distributed Control Architecture for an Autonomous Mobile Robot*', 1986, Artificial Intelligence, p. 99-108.

54. Engelmore, R. and A. Terry: '*Structure and Function of the CRYSALIS System*', 1979, IOCAI, p. 250-256.

55. Engelmore, R. and T. Morgan: '*Blackboard Systems*', 1988, Addison-Wesley.

56. Ensor, J.R. and J.D. Gabbe: '*Transactional Blackboards*', 1986, Artificial Intelligence, p. 80-84.

57. Erman, L.D. and V.R. Lesser: '*A Multi-Level Organization For Problem Solving Using Many, Diverse, Cooperating Sources Of Knowledge*', 1975, in *IJCAI 4*, Tbilisi, USSR, p. 483-490.

58. Erman, L.D., P.E. London, and S.F. Fickas: '*The Design and an Example use of Hearsay III*', 1979, IOCAI, p. 409-415.

59. Erman, L.D., *et al.*: '*The Hearsay Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty*', 1980, Computing Surveys, p. 213-253.

60. Forgy, C.L.: '*Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*', 1982, Artificial Intelligence, p. 17-37.

61. Gallagher, K.Q. and D.D. Corkill: '*Performance Aspects of GBB*', 1989, in *Blackboard Architectures and Applications*, V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, p. 323-346.

62. Gillies, A.C.: '*The Integration of Expert Systems Into Mainstream Software*', 1991, Chapman & Hall Computing.

63. Gilmore, J.F., S.P. Roth, and S.D. Tynor: '*A Blackboard System for Distributed Problem Solving*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 371-393.

64. Hall, G.A. and J.T. Thompson: '*Satellite Control Using Ada and Expert Systems*', 1988, in *AIDA:The 4th Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 7-1 to 7-10.

65. Harding, D.S. and K.L. Albin: '*Taking Inference to Task*', 1988, in *AIDA:The 4th Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 8-1 to 8-9.

66. Harland, D.M.: '*REKURSIV object-oriented computer architecture*', 1988, Ellis Horwood.

67. Harrison, A. and P. Thomas: '*Knowledge Representation in Real-Time Embedded Systems*', 1992, in *IEE Colloquium "Real-Time Knowledge-Based Systems,* IEE, Savoy Place, London, p. 2/1 - 2/3.

68. Hasse, V.H.: '*The Use of AI Methods in the Implementation of Real-Time Software Products*', 1990, in *Automatic Control,* Tallin, USSR, p. 247 - 251.

69. Hayes-Roth, B. and F. Hayes-Roth: '*Modelling Planning as an Incremental, Opportunistic Process*', 1979, IOCAI, p. 375-363.

70. Hayes-Roth, B.:'*A Blackboard Model of Control*', 1984, Stanford University.

71. Hayes-Roth, B.: '*A Blackboard Architecture for Control*', 1985, Artificial Intelligence, p. 251-321.

72. Hayes-Roth, B., *et al.*: '*Application of the BB1 blackboard control architecture to arrangement-assembly tasks*', 1986, Artificial Intelligence, p. 85-94.

73. Hayes-Roth, B.: '*Control in Blackboard Systems*', 1989, in *Blackboard Architectures and Applications,* Jagannathan.V, R. Dodhiawala, and L.S. Baum, p. xv-xvii.

74. Hayslip, I.C. and J.P. Rosenking: '*Adaptive Planning for Threat Response*', 1989, in *Applications of Artificial Intelligence,* p. 1031-1041.

75. Hewett, M. and B. Hayes-Roth: '*Real-Time I/O in Knowledge-Based Systems*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 269-283.

76. Hintz, J.C.: '*Using Ada in Expert System Development*', 1991, in *Test Engineering Conference,* USA, p. 545-558.

77. Hirshfield, H.H. and T.B. Slack: '*ERS: An Expert System Shell Designed and Implemented in Ada*', 1988, in *AIDA:The 4th Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 10-1 to 10-34.

78. Ho, C.: '*Explicit Context-Based Blackboards Enhancing Blackboard Systems Performance*', 1989, 4th Portuguese Conference on Artificial Intelligence, p. 73-84.

79. Ice, S., *et al.*: '*Raising ALLAN: Ada Logic-Based Language*', 1987, in *AIDA:The 3nd Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 155-165.

80. Ichbiah, J.D.: '*Reference Manual for the Ada Programming Language*', 1983, Castle House Publications Ltd.

81. Jagannathan, V., L.S. Baum, and R.T. Dodhiawala: '*Erasmus: Reconfiguarable Object Oriented Blackboard Systems*', 1987, Methodologies for Intelligent Systems, p. 87-95.

82. Jagannathan, V.: '*Realizing the Concurrent Blackboard Model*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press, p. 85-87.

83. Jaworski, A., D. LaVallee, and D. Zoch: '*A LISP-Ada Connection for Expert System Development*', 1987, in *AIDA:The 3rd Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 74-89.

84. Johnson, P.M., D.D. Corkill, and K.Q. Gallagher: '*Integrating BB1-Style Control into the Generic Blackboard System*', 1987, in *Workshop on Blackboard Systems,* Seatle, p. 1-14.

85. Johnson, M.V. and B. Hayes-Roth: '*Simultaneous Dynamic Integration of Diverse Reasoning Methods*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press, p. 57-73.

86. Junqua, J.: '*A Knowledge Engineering Approach to Speech Processing Using a Blackboard Model: Application to Automatic Labelling*', 1989, Engineering Applications of AI, p. 198-206.

87. Kaiser, K., *et al.*: '*Adapting the Blackboard Model for Cockpit Information Management*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.D. Dodhiawala, and L.S. Baum, Academic Press, p. 481-500.

88. Kilpelainen, P., *et al.*: '*Prolog in Ada: An Implementation and an Embedding*', 1989, in *AIDA-89:The 5th Annual Conference on Artificial Intelligence and Ada,* George Mason University, USA, p. 96-107.

89. Kimble, C.: '*Gaining Control of Reasoning Through Ada Designed AI Tools for Real-Time*', 1990, in *AIDA -90, The Sixth Anual Conference on Artificial Intelligence in Ada,* Reston, Verginia, USA, p. 70-84.

90. Kluzniak, F., S. Szpakowics, and J.S. Bien: '*Prolog for Programmers*', 1985, Academic-Press.

91. Kunz, J.C., T.P. Kehler, and M.D. Williams: '*Applications Development Using a Hybrid AI Development System*', 1984, p. 41-54.

92. Kuru, S. and F. Bek: '*Goal-Driven Blackboard Control Architecture Based on Extending Partially Complete General Goal Trees*', 1990, Knowledge Based Systems, p. 236-241.

93. Laasri, H., B. Maitre, and J.P. Haton: '*Organization, Cooperation and Exploitation of Knowledge in Blackboard Architectures: ATOME*', 1988, in *Expert Systems and Their Applications,* Avignon(France), p. 371-390.

94. Laasri, H. and B. Maitre: '*Flexibility and Efficiency in Blackboard Systems:Studies and Achievements in ATOME*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 309-322.

95. Labhart, J. and K. Williams: '*Ada MeriTool: A Software Tool For Knowledge-Based Systems*', 1989, in *AIDA:The 5th Annual Conference Artificial Intelligence and Ada,* George Mason University, USA, p. 63-73.

96. Lalanda, P., F. Charpillet, and J. Haton: '*A Real-Time Blackboard-Based Architecture*', 1992, in *ECAI,* Vienna, p. 262-266.

97. Land, L. and T. Mulhall: '*Developing Cooperative Knowledge-Based Systems*', 1989, in *Expert Systems,* p. 90 - 103.

98. Leao, L.V. and S.N. Talukdar: '*An Environment For Rule-Based Blackboards and Distributed Problem Solving*', 1986, Artificial Intelligence, p. 70-79.

99. Lesser, V.R. and D.D. Corkill: '*The Distributed Vehicle Monitoring Testbed: A Tool For Investigating Distributed Problem Solving Networks*', 1883, The AI Magazine, p. 15-33.

100. Lesser, V.R., *et al.*: '*Goal Relationships and Their Use in a Blackboard Architecture*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press, p. 9-26.

101. Lippolt, B., H. Velthuijsen, and J. Vonk: '*A blackboard shell with multiple blackboards*', 1987, in *KBS 87*, p. 71-78.

102. Lucas, P. and L. VanDerGaag: '*Principles of Expert Systems*', 1991, Addison-Wesley.

103. Marik, V., L. Lhotska, and I. Sediva: '*Blackboard Control Structure in the FEL-Expert System*', 1988, IFAC/IMACS Symposium, p. 187-192.

104. Marik, V. and L. Lhotska: '*Some Aspects of the Blackboard Control in the Diagnostic FEL-Expert Shell*', 1989, Artificial Intelligence and Information-Control Systems of Robots, p. 179-182.

105. Mason, K.P. and S.T. Hodd: '*KBESM-A Prolog Blackboard System*', 1989, The Australian Computer Journal, p. 100-107.

106. Miles, J.A.H.: '*Artificial Intelligence Applied to Data Fusion and Situation Assessment for Command and Control*', 1988, University of Southampton.

107. Miles, J.A.H.: '*Architectures for C2 Knowledge-Based Systems*', 1989, Third International Conference on C3MIS, p. 64-69.

108. Milzner, K.: '*An Analog Design Environment Based on Cooperating Blackboard Systems*', 1991, Applied Intelligence, p. 179-194.

109. Minsky, M.: '*A framework for representing knowledge*', 1975, in *The Psychology of Computer Vision*, P. Winston H., McGraw-Hill, p. 211-277.

110. Murray, W.R.: '*Dynamic Instructional Planning in the BB1 Blackboard Architecture*', 1989, in *Blackboard Architectures and Applications*, V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press, p. 455-480.

111. Naedal, D.: '*Ada and Embedded AI*', 1986, Defense`Electronics, p. 90-100.

112. Nii, H.P. and E.A. Feigenbaum: '*Rule-Based Understanding of Signals*', 1978, in *Pattern-Directed Inference Systems,* D.A. Waterman and B. Hayes-Roth, Academic Press, p. 483-501.

113. Nii, H.P.:'*An Introduction to Knowledge Engineering, Blackboard Model, and AGE*', 1980, DARPA.

114. Nii, H.P., *et al.*: '*Signal-to-Symbol Transformation: HASP/SIAP Case Study*', 1982, The AI Magazine, p. 23-35.

115. Nii, H.P.: '*Blackboard Systems:The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures*', 1986, The AI Magazine, p. 38-52.

116. Nii, H.P.: '*Blackboard Systems:Blackboard Application Systems, Blackboard Systems from a Knowledge Engineering Perspective*', 1986, The AI Magazine, p. 82-104.

117. Nii, H.P.: '*Introduction - Blackboard Architectures and Applications*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press,

118. Nilsson, J.N.: '*Principles of Artificial Intelligence*', 1980, Springer-Verlag.

119. Partridge, D.: '*Engineering Artificial Intelligence Software*', 1992, Intellect.

120. Pearson, G.: '*A Blackboard System for Planning Space Missions*', 1989, in *Industrial Applications of Artificial Intelligence and Expert Systems,* p. 409 - 416.

121. Porquet, C., M. Desvignes, and M. Revenu: '*A Blackboard-Based Architecture for the Interpretation of Image Sequences*', 1990, in *SPIE Intelligent Robots and Computer Vision,* p. 709-720.

122. Raulefs, P.: '*Toward a Blackboard Architecture for Real-Time Interactions with Dynamic Systems*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R.T. Dodhiawala, and L.S. Baum, Academic Press, p. 285-289.

123. Reichgelt, H.: '*Knowledge Representation An AI Perspective*', 1991, Ablex Publishing Corporation.

124. Rice, J., N. Aiello, and H.P. Nii: '*See How They Run...The Architecture and Performance of Two Concurrent Blackboard Systems*', 1989, in *Blackboard Architectures and Applications,* V. Jagannathan, R. Dodhiawala, and L.S. Baum, Academic Press, p. 153-178.

125. Rich, E.: '*Artificial Intelligence*', 1986, McGraw-Hill.

126. Ringland, G.A. and D.A. Duce: '*Approaches to Knowledge Representation - An Introduction*', 1988, Research Studies Press.

127. Robinson, J.A.: '*A Machine Oriented Logic Based on the Resolution Principle*', 1965, Journal of the Association for Computing Machinery, p. 23-41.

128. Roth, A.: '*The Practical Applications of Prolog*', 1994.

129. Saxena, M.K.: '*Blackboard Architecture-A Tutorial Introduction.*', 1988, Institute of Electronics & Telecommunication Engineers, p. 198-208.

130. Saxena, M.K.: '*Knowledge Representation in Distributed Blackboard Architectures - Some Issues*', 1989, in *Knowledge-Based Computer Systems,* Bombay, p. 230-239.

131. Scheidt, D., D. Preston, and M. Armstrong: '*Implementing Semantic Networks in Ada*', 1986, in *AIDA-86:The 2nd Annual Conference on Artificial Intelligence and Ada,* George Mason University, USA, p. 8-1 to 8-16.

132. Schwartz, R.L. and P.M. Melliar-Smith:'*The Suitability of Ada for Artificial Intelligence'*, 1980, SRI International.

133. Scoy, V. and S.M. Reddy: '*Ada for Knowledge Representation: Lessons Learned'*, 1987, in *AIDA-87*, George Mason University, USA, p. 90 - 101.

134. Sibley, E.H.: '*Ada Policy: Yesterday, Today and Tomorrow'*, 1986, AIDA: The 2nd Annual Conference on Artificial Intelligence and Ada, p. 1-8.

135. Siddiqi, J.I.A., J.H. Sumiga, and B. Khazaei: '*Use of a Blackboard Framework to Model Software Design'*, 1988, Empirical Foundations of Information and Software Science, p. 99-107.

136. Silverman, B.G., J.S. Chang, and K. Feggos: '*Blackboard System Generator (BSG): An Alternative Distributed Problem-Solving Paradigm'*, 1989, IEEE Transactions on Systems, Man, and Cybernetics, p. 334-355.

137. Sloman, A.: '*Why We Need Many Knowledge Representation Formalisms'*, 1985, in *Research and Development in Expert Systems*, p. 163 - 183.

138. Stockman, S.P.: '*ABLE: An Ada-Based Blackboard System'*, 1988, In *AIDA:The 4th Annual Conference Artificial Intelligence and Ada*, George Mason University USA, p. 18-1 to 18-9.

139. Talukdar, S.N., E. Cardozo, and L.V. Leao: '*TOAST:The Power System Operator's Assistant'*, 1986, IEEE Computer, p. 53-60.

140. Terry, A.:'*The CRYSALIS Project: Hierarchical Control of Production Systems'*, 1983, Stanford University.

141. Thomas, P., H. Robinson, and J. Emms: '*Abstract Data Types:their specification, representation and use.'*, 1988, Oxford University Press.

142. Tyler, S.W., *et al.*: '*CHORIS: Integrating Multiple Experts to Support Intelligent Interfaces & Embedded Training'*, 1991, p. 1 - 27.

143. Venkatasubramanian, V. and C. Chen: '*A Blackboard Architecture for Plastics Design*', 1986, Artificial Intelligence, p. 117-122.

144. Wallnau, K.C., *et al.*: '*Construction of Knowledge-Based Components and Applications in Ada*', 1988, in *AIDA-88:The 4th Annual Conference on Artificial Intelligence and Ada*, George Mason University, USA, p. 3-1 to 3-21.

145. Wilber, G.F.: '*Intelligent Real-Time Embedded Systems*', 1989, in *AIDA:The 5th Annual Conference Artificial Intelligence and Ada*, George Mason University, USA, p. 74-82.

146. Williams, K.A.: '*Ada: A Language for Artificial Intelligence*', 1987, in *3rd Artificial Intelligence and Advanced Computer Technology Conference*, p. 98-110.

147. Winston, P.H.: '*Artificial Intelligence*', 1992, Addison-Wesley.

148. Wirth, N.: '*Algorithms + Data Structures = Programs*', 1976, Prentice-Hall.

149. Woods, D.: '*Space Station Freedom:Embedding AI*', 1992, AI Expert, p. 33-39.

150. Wright, P.A.: '*Ada Real-Time Inference Engine (ARTIE)*', 1989, in *AIDA-89*, AIDA-89:The 5th Annual Conference on Artificial Intelligence and Ada, p. 83-93.

151. Yoshida, N. and K. Hino: '*An Object-Oriented Framework of Pattern Recognition Systems*', 1988, OOPSLA 1988, p. 259-266.

UNRESTRICTED

The Integration of Multiple and Diverse Knowledge Representation Paradigms using a Blackboard Architecture

Annex A

Abstract Data Types & Abstract Knowledge Types

A Thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy in Computer Science

Alan Harrison BA MSc

Faculty of Mathematics and Computing
The Open University

December 1994

Volume 2 of 3

Contents

Abstract Data Types

Abstract Knowledge Types

Abstract Data Types

Free List

```
------------------------------------------------------------------------
--
-- Unit      : FREE_LIST_PACKAGE specification
-- Name      : A Harrison Software Engineering Group,
--             Cranfield University,
--             RMCS, Shrivenham
-- Date      : September 1990
-- Version   : 1
-- Function : Garbage collector specification for dynamic string
--
------------------------------------------------------------------------
generic
   type ITEM is limited private;
   type POINTER is access ITEM;

   with procedure FREE(
   THE_ITEM  : in out  ITEM);

   with procedure SET_POINTER(
        THE_ITEM       : in out  ITEM;
        THE_POINTER    : in        POINTER);

   with function POINTER_OF(
        THE_ITEM  : in ITEM) return POINTER;

------------------------------------------------------------------------
package FREE_LIST_PACKAGE is
------------------------------------------------------------------------

   procedure FREE(
   THE_POINTER : in out POINTER);

   function  NEW_ITEM    return POINTER;

------------------------------------------------------------------------
end FREE_LIST_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : FREE_LIST_PACKAGE body
-- Name      : A Harrison Software Engineering Group,
--             Cranfield University,
--             RMCS, Shrivenham
-- Date      : Sep 1990
-- Version   : 1
-- Function : Garbage collector body  for dynamic string
--
-------------------------------------------------------------------
package body FREE_LIST_PACKAGE is
-------------------------------------------------------------------
   FREE_LIST : POINTER := null;
   ----------------------------------------------------------------
   procedure FREE(
   ----------------------------------------------------------------
   THE_POINTER : in out POINTER) is
   ----------------------------------------------------------------
     TEMPORARY_POINTER : POINTER;
   ----------------------------------------------------------------
   begin
     while THE_POINTER /= null
     loop
       TEMPORARY_POINTER := THE_POINTER;
       THE_POINTER := POINTER_OF(THE_POINTER.all);
       FREE(TEMPORARY_POINTER.all);
       SET_POINTER(TEMPORARY_POINTER.all, THE_POINTER => FREE_LIST);
       FREE_LIST := TEMPORARY_POINTER;
     end loop;
   ----------------------------------------------------------------
   end FREE;
   ----------------------------------------------------------------


  ----------------------------------------------------------------
   function NEW_ITEM return POINTER is
   ----------------------------------------------------------------
     TEMPORARY_POINTER : POINTER;
   ----------------------------------------------------------------
   begin
     if FREE_LIST = null then
       return new ITEM;
     else
       TEMPORARY_POINTER := FREE_LIST;
       FREE_LIST := POINTER_OF(TEMPORARY_POINTER.all);
       SET_POINTER(TEMPORARY_POINTER.all, THE_POINTER => null);
       return TEMPORARY_POINTER;
     end if;
   ----------------------------------------------------------------
   end NEW_ITEM;
   ----------------------------------------------------------------

-------------------------------------------------------------------
end FREE_LIST_PACKAGE;
-------------------------------------------------------------------
```

Dynamic String

```
------------------------------------------------------------------------
--
-- Unit      : DYNAMIC_STRING_PACKAGE specification
-- Name      : A Harrison Software Engineering Group,
--             Cranfield University,
--             RMCS, Shrivenham
-- Date      : Sep 1990
-- Version   : 1
-- Function  : Dynamic string specification
--             Booch "Software Components in Ada"
--
------------------------------------------------------------------------
generic
   type ITEM is private;
   type SUBSTRING is array(POSITIVE range <>) of ITEM;

   with function "<"  ( LEFT   : in ITEM;
                        RIGHT  : in ITEM)  return BOOLEAN;

   with procedure PUT    ( THE_SUBSTRING : in SUBSTRING);

------------------------------------------------------------------------
package DYNAMIC_STRING_PACKAGE IS
------------------------------------------------------------------------

   type STRING is private;

   procedure PUT(
   THE_STRING            : in      STRING);

   procedure COPY(
   FROM_THE_STRING       : in      STRING;
   TO_THE_STRING         : in out  STRING);

   procedure COPY(
   FROM_THE_SUBSTRING    : in      SUBSTRING;
   TO_THE_STRING         : in out  STRING);

   procedure CLEAR(
   THE_STRING            : in out  STRING);

   procedure PREPEND(
   THE_STRING            : in      STRING;
   TO_THE_STRING         : in out  STRING);

   procedure PREPEND(
   THE_SUBSTRING         : in      SUBSTRING;
   TO_THE_STRING         : in out  STRING);

   procedure PREPEND(
   THE_ITEM              : in      ITEM;
   TO_THE_STRING         : in out  STRING);

   procedure APPEND(
   THE_STRING            : in      STRING;
   TO_THE_STRING         : in out  STRING);
```

```
procedure APPEND(
THE_SUBSTRING          : in      SUBSTRING;
TO_THE_STRING          : in out  STRING);

procedure APPEND(
THE_ITEM               : in      ITEM;
TO_THE_STRING          : in out  STRING);

procedure INSERT(
THE_STRING             : in      STRING;
IN_THE_STRING          : in out  STRING;
AT_THE_POSITION        : in      POSITIVE);

procedure INSERT(
THE_SUBSTRING          : in      SUBSTRING;
IN_THE_STRING          : in out  STRING;
AT_THE_POSITION        : in      POSITIVE);

procedure DELETE(
IN_THE_STRING          : in out  STRING;
FROM_THE_POSITION      : in      POSITIVE;
TO_THE_POSITION        : in      POSITIVE);

procedure REPLACE(
IN_THE_STRING          : in out  STRING;
AT_THE_POSITION        : in      POSITIVE;
WITH_THE_STRING        : in      STRING);

procedure REPLACE(
IN_THE_STRING          : in out  STRING;
AT_THE_POSITION        : in      POSITIVE;
WITH_THE_SUBSTRING     : in      SUBSTRING);

procedure SET_ITEM(
IN_THE_STRING          : in out  STRING;
AT_THE_POSITION        : in        POSITIVE;
WITH_THE_ITEM          : in        ITEM);

function COPY(
FROM_THE_STRING        : in STRING)        return STRING;

function IS_EQUAL(
LEFT                   : in STRING;
RIGHT                  : in STRING)        return BOOLEAN;

function IS_EQUAL(
LEFT                   : in SUBSTRING;
RIGHT                  : in STRING)        return BOOLEAN;

function IS_EQUAL(
LEFT                   : in STRING;
RIGHT                  : in SUBSTRING)     return BOOLEAN;

function IS_LESS_THAN(
LEFT                   : in STRING;
RIGHT                  : in STRING)        return BOOLEAN;

function IS_LESS_THAN(
LEFT                   : in SUBSTRING;
RIGHT                  : in STRING)        return BOOLEAN;
```

```
   function IS_LESS_THAN(
   LEFT                  : in STRING;
   RIGHT                 : in SUBSTRING)    return BOOLEAN;

   function IS_GREATER_THAN(
   LEFT                  : in STRING;
   RIGHT                 : in STRING)       return BOOLEAN;

   function IS_GREATER_THAN(
   LEFT                     : in SUBSTRING;
   RIGHT                 : in STRING)       return BOOLEAN;

   function IS_GREATER_THAN(
   LEFT                  : in STRING;
   RIGHT                 : in SUBSTRING)    return BOOLEAN;

   function LENGTH_OF(
   THE_STRING            : in STRING)       return NATURAL;

   function IS_NULL(
   THE_STRING            : in STRING)       return BOOLEAN;

   function ITEM_OF(
   THE_STRING            : in STRING;
   AT_THE_POSITION       : in POSITIVE)     return ITEM;

   function SUBSTRING_OF(
   THE_STRING            : in STRING)       return SUBSTRING;

   function SUBSTRING_OF(
   THE_STRING            : in STRING;
   FROM_THE_POSITION     : in POSITIVE;
   TO_THE_POSITION       : in POSITIVE)     return SUBSTRING;

   OVERFLOW          : EXCEPTION;
   POSITION_ERROR    : EXCEPTION;

private

   type STRUCTURE is access SUBSTRING;
   type STRING is
     record
       THE_LENGTH  : NATURAL := 0;
       THE_ITEMS   : STRUCTURE;
     end record;

-------------------------------------------------------------------------
end DYNAMIC_STRING_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : DYNAMIC_STRING_PACKAGE body
-- Name      : A Harrison Software Engineering Group, Cranfield
--             University, RMCS, Shrivenham
-- Date      : Sep 1990
-- Version   : 1
-- Function : Dynamic string body Booch "Software Components in Ada"
--
-------------------------------------------------------------------
with FREE_LIST_PACKAGE, TEXT_IO;
-------------------------------------------------------------------
package body DYNAMIC_STRING_PACKAGE is
-------------------------------------------------------------------

   type NODE;
   type NODE_POINTER is access NODE;
   type NODE is
      record
         THE_STRUCTURE : STRUCTURE;
         NEXT          : NODE_POINTER;
      end record;

   type HEADER;
   type HEADER_POINTER is access HEADER;
   type HEADER is
      record
         THE_SIZE       : NATURAL;
         THE_STRUCTURES : NODE_POINTER;
         NEXT           : HEADER_POINTER;
   end record;

   FREE_LIST : HEADER_POINTER;

   -------------------------------------------------------------------
   procedure FREE(
   -------------------------------------------------------------------
   THE_NODE : in out NODE) is
   -------------------------------------------------------------------
   begin
      THE_NODE.THE_STRUCTURE := null;
   -------------------------------------------------------------------
   end FREE;
   -------------------------------------------------------------------


   -------------------------------------------------------------------
   procedure SET_NEXT(
   -------------------------------------------------------------------
   THE_NODE    : in out NODE;
   TO_NEXT     : in        NODE_POINTER) is
   -------------------------------------------------------------------
   begin
      THE_NODE.NEXT := TO_NEXT;
   -------------------------------------------------------------------
   end SET_NEXT;
   -------------------------------------------------------------------
```

```
-------------------------------------------------------------------
function NEXT_OF(
-------------------------------------------------------------------
THE_NODE : in NODE) return NODE_POINTER is
-------------------------------------------------------------------
begin
   return THE_NODE.NEXT;
-------------------------------------------------------------------
end NEXT_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
package NODE_MANAGER is new FREE_LIST_PACKAGE(
-------------------------------------------------------------------
ITEM           => NODE,
POINTER        => NODE_POINTER,
FREE           => FREE,
SET_POINTER    => SET_NEXT,
POINTER_OF     => NEXT_OF);
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure FREE(
-------------------------------------------------------------------
THE_HEADER : in out HEADER) is
-------------------------------------------------------------------
begin
   THE_HEADER.THE_SIZE := 0;
-------------------------------------------------------------------
end FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_NEXT(
-------------------------------------------------------------------
THE_HEADER     : in out  HEADER;
TO_NEXT      : in      HEADER_POINTER) is
-------------------------------------------------------------------
begin
   THE_HEADER.NEXT := TO_NEXT;
-------------------------------------------------------------------
end SET_NEXT;
-------------------------------------------------------------------


-------------------------------------------------------------------
function NEXT_OF(
-------------------------------------------------------------------
THE_HEADER : in HEADER) return HEADER_POINTER is
-------------------------------------------------------------------
begin
   return THE_HEADER.NEXT;
-------------------------------------------------------------------
end NEXT_OF;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
package HEADER_MANAGER is new FREE_LIST_PACKAGE(
-------------------------------------------------------------------
ITEM            => HEADER,
POINTER         => HEADER_POINTER,
FREE            => FREE,
SET_POINTER     => SET_NEXT,
POINTER_OF      => NEXT_OF);
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure FREE(
-------------------------------------------------------------------
THE_STRUCTURE : in out STRUCTURE) is
-------------------------------------------------------------------
   NODE_INDEX       : NODE_POINTER;
   PREVIOUS_HEADER  : HEADER_POINTER;
   HEADER_INDEX     : HEADER_POINTER := FREE_LIST;
-------------------------------------------------------------------
begin
   while HEADER_INDEX /= null loop
      if THE_STRUCTURE'LENGTH < HEADER_INDEX.THE_SIZE then
         exit;
      elsif THE_STRUCTURE'LENGTH = HEADER_INDEX.THE_SIZE then
         NODE_INDEX := NODE_MANAGER.NEW_ITEM;
         NODE_INDEX.THE_STRUCTURE := THE_STRUCTURE;
         NODE_INDEX.NEXT := HEADER_INDEX.THE_STRUCTURES;
         HEADER_INDEX.THE_STRUCTURES := NODE_INDEX;
         THE_STRUCTURE := null;
         return;
      end if;
      PREVIOUS_HEADER := HEADER_INDEX;
      HEADER_INDEX := HEADER_INDEX.NEXT;
   end loop;
   HEADER_INDEX := HEADER_MANAGER.NEW_ITEM;
   HEADER_INDEX.THE_SIZE := THE_STRUCTURE'LENGTH;
   NODE_INDEX := NODE_MANAGER.NEW_ITEM;
   NODE_INDEX.THE_STRUCTURE := THE_STRUCTURE;
   HEADER_INDEX.THE_STRUCTURES := NODE_INDEX;
   if PREVIOUS_HEADER = null THEN
      HEADER_INDEX.NEXT := FREE_LIST;
      FREE_LIST := HEADER_INDEX;
   else
      HEADER_INDEX.NEXT := PREVIOUS_HEADER.NEXT;
      PREVIOUS_HEADER.NEXT := HEADER_INDEX;
   end if;
   THE_STRUCTURE := null;
-------------------------------------------------------------------
end FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
function NEW_STRUCTURE(
-------------------------------------------------------------------
THE_SIZE : in NATURAL) return STRUCTURE is
-------------------------------------------------------------------
   THE_STRUCTURE    : STRUCTURE;
   NODE_INDEX       : NODE_POINTER;
```

```
   PREVIOUS_HEADER : HEADER_POINTER;
   HEADER_INDEX    : HEADER_POINTER := FREE_LIST;
-------------------------------------------------------------------
begin
   while HEADER_INDEX /= null
   loop
      if HEADER_INDEX.THE_SIZE = THE_SIZE then
      -- Note change from Booch in order to make all strings the
      -- exact length
         NODE_INDEX := HEADER_INDEX.THE_STRUCTURES;
         HEADER_INDEX.THE_STRUCTURES := NODE_INDEX.NEXT;
         NODE_INDEX.NEXT := null;
         if HEADER_INDEX.THE_STRUCTURES = null then
            if PREVIOUS_HEADER = null then
               FREE_LIST := HEADER_INDEX.NEXT;
            else
               PREVIOUS_HEADER.NEXT := HEADER_INDEX.NEXT;
            end if;
            HEADER_INDEX.NEXT := null;
            HEADER_MANAGER.FREE(HEADER_INDEX);
         end if;
         THE_STRUCTURE := NODE_INDEX.THE_STRUCTURE;
         NODE_MANAGER.FREE(NODE_INDEX);
         return THE_STRUCTURE;
      end if;
      PREVIOUS_HEADER := HEADER_INDEX;
      HEADER_INDEX := HEADER_INDEX.NEXT;
   end loop;
   return new SUBSTRING(1..THE_SIZE);
-------------------------------------------------------------------
end NEW_STRUCTURE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET(
-------------------------------------------------------------------
THE_STRING           : in out  STRING;
TO_THE_SIZE          : in      NATURAL;
PRESERVE_THE_VALUE   : in      BOOLEAN) is
-------------------------------------------------------------------
   TEMPORARY_STRUCTURE : STRUCTURE;
-------------------------------------------------------------------
begin
   if TO_THE_SIZE = 0 then
      FREE(THE_STRING.THE_ITEMS);
   elsif THE_STRING.THE_ITEMS = null then
      THE_STRING.THE_ITEMS :=   NEW_STRUCTURE
                                (THE_SIZE => TO_THE_SIZE);
   elsif TO_THE_SIZE > THE_STRING.THE_ITEMS'LENGTH then
      if PRESERVE_THE_VALUE then
         TEMPORARY_STRUCTURE := NEW_STRUCTURE
                                (THE_SIZE => TO_THE_SIZE);
         TEMPORARY_STRUCTURE(1..THE_STRING.THE_LENGTH) :=
           THE_STRING.THE_ITEMS(1..THE_STRING.THE_LENGTH);
         FREE(THE_STRING.THE_ITEMS);
         THE_STRING.THE_ITEMS := TEMPORARY_STRUCTURE;
      else
         FREE(THE_STRING.THE_ITEMS);
         THE_STRING.THE_ITEMS := NEW_STRUCTURE
```

```
                        (THE_SIZE => TO_THE_SIZE);
    end if;
  end if;
  THE_STRING.THE_LENGTH := TO_THE_SIZE;
-------------------------------------------------------------------
end SET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PUT(
-------------------------------------------------------------------
THE_STRING : in STRING) is
-------------------------------------------------------------------
begin
  PUT(SUBSTRING_OF(STRING(THE_STRING)));
-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure COPY(
-------------------------------------------------------------------
FROM_THE_STRING    : in       STRING;
TO_THE_STRING      : in out   STRING) is
-------------------------------------------------------------------
begin
  SET(TO_THE_STRING,
      TO_THE_SIZE          => FROM_THE_STRING.THE_LENGTH,
      PRESERVE_THE_VALUE   => FALSE);
  TO_THE_STRING.THE_ITEMS(1..FROM_THE_STRING.THE_LENGTH) :=
    FROM_THE_STRING.THE_ITEMS(1..FROM_THE_STRING.THE_LENGTH);
-------------------------------------------------------------------
exception
  when STORAGE_ERROR =>
    raise OVERFLOW;
-------------------------------------------------------------------
end COPY;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure COPY(
-------------------------------------------------------------------
FROM_THE_SUBSTRING   : in     SUBSTRING;
TO_THE_STRING        : in out STRING) is
-------------------------------------------------------------------
begin
  SET(TO_THE_STRING,
      TO_THE_SIZE          => FROM_THE_SUBSTRING'LENGTH,
      PRESERVE_THE_VALUE   => FALSE);
  TO_THE_STRING.THE_ITEMS(1..FROM_THE_SUBSTRING'LENGTH) :=
    FROM_THE_SUBSTRING;
-------------------------------------------------------------------
exception
  when STORAGE_ERROR =>
    raise OVERFLOW;
-------------------------------------------------------------------
end COPY;
```

```
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure CLEAR(
-------------------------------------------------------------------
THE_STRING       : in out  STRING) is
-------------------------------------------------------------------
begin
   SET(THE_STRING,
        TO_THE_SIZE           => 0,
        PRESERVE_THE_VALUE    => FALSE);
-------------------------------------------------------------------
end CLEAR;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PREPEND(
-------------------------------------------------------------------
THE_STRING       : in      STRING;
TO_THE_STRING    : in out  STRING) is
-------------------------------------------------------------------
   OLD_LENGTH : NATURAL :=   TO_THE_STRING.THE_LENGTH;
   NEW_LENGTH : NATURAL :=   TO_THE_STRING.THE_LENGTH +
                             THE_STRING.THE_LENGTH;
-------------------------------------------------------------------
begin
   SET(TO_THE_STRING,
        TO_THE_SIZE           => NEW_LENGTH,
        PRESERVE_THE_VALUE    => TRUE);
   TO_THE_STRING.THE_ITEMS((THE_STRING.
                            THE_LENGTH + 1)..NEW_LENGTH) :=
   TO_THE_STRING.THE_ITEMS(1..OLD_LENGTH);
   TO_THE_STRING.THE_ITEMS(1..THE_STRING.THE_LENGTH) :=
     THE_STRING.THE_ITEMS(1..THE_STRING.THE_LENGTH);
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
      raise OVERFLOW;
-------------------------------------------------------------------
end PREPEND;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PREPEND(
-------------------------------------------------------------------
THE_SUBSTRING    : in      SUBSTRING;
TO_THE_STRING    : in out  STRING) is
-------------------------------------------------------------------
   OLD_LENGTH : NATURAL :=   TO_THE_STRING.THE_LENGTH;
   NEW_LENGTH : NATURAL :=   TO_THE_STRING.THE_LENGTH +
                             THE_SUBSTRING'LENGTH;
-------------------------------------------------------------------
begin
   SET(TO_THE_STRING,
        TO_THE_SIZE           => NEW_LENGTH,
        PRESERVE_THE_VALUE    => TRUE);
   TO_THE_STRING.THE_ITEMS((THE_SUBSTRING'LENGTH + 1)..
```

```
                                NEW_LENGTH) :=
    TO_THE_STRING.THE_ITEMS(1..OLD_LENGTH);
    TO_THE_STRING.THE_ITEMS(1..THE_SUBSTRING'LENGTH) :=
                                                    THE_SUBSTRING;
  ------------------------------------------------------------------
  exception
    when STORAGE_ERROR =>
      raise OVERFLOW;
  ------------------------------------------------------------------
  end PREPEND;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure PREPEND(
  ------------------------------------------------------------------
  THE_ITEM         : in     ITEM;
  TO_THE_STRING    : in out STRING) is
  ------------------------------------------------------------------
    OLD_LENGTH : NATURAL := TO_THE_STRING.THE_LENGTH;
    NEW_LENGTH : NATURAL := TO_THE_STRING.THE_LENGTH + 1;
  ------------------------------------------------------------------
  begin
    SET(TO_THE_STRING,
        TO_THE_SIZE        => NEW_LENGTH,
        PRESERVE_THE_VALUE   => TRUE);
    TO_THE_STRING.THE_ITEMS(2 .. NEW_LENGTH) :=
      TO_THE_STRING.THE_ITEMS(1..OLD_LENGTH);
    TO_THE_STRING.THE_ITEMS(1) := THE_ITEM;
  ------------------------------------------------------------------
  exception
    when STORAGE_ERROR =>
      raise OVERFLOW;
  ------------------------------------------------------------------
  end PREPEND;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure APPEND(
  ------------------------------------------------------------------
  THE_STRING       : in     STRING;
  TO_THE_STRING    : in out STRING) is
  ------------------------------------------------------------------
    OLD_LENGTH : NATURAL :=   TO_THE_STRING.THE_LENGTH;
    NEW_LENGTH : NATURAL :=   TO_THE_STRING.THE_LENGTH +
                              THE_STRING.THE_LENGTH;
  ------------------------------------------------------------------
  begin
    SET(TO_THE_STRING,
        TO_THE_SIZE        => NEW_LENGTH,
        PRESERVE_THE_VALUE   => TRUE);
    TO_THE_STRING.THE_ITEMS((OLD_LENGTH + 1) .. NEW_LENGTH)
      := THE_STRING.THE_ITEMS(1 .. THE_STRING.THE_LENGTH);
  ------------------------------------------------------------------
  exception
    when STORAGE_ERROR =>
      raise OVERFLOW;
  ------------------------------------------------------------------
  end APPEND;
```

```
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure APPEND(
-------------------------------------------------------------------
THE_SUBSTRING    : in        SUBSTRING;
TO_THE_STRING    : in out  STRING) is
-------------------------------------------------------------------
   OLD_LENGTH :  NATURAL :=  TO_THE_STRING.THE_LENGTH;
   NEW_LENGTH :  NATURAL :=  TO_THE_STRING.THE_LENGTH +
                             THE_SUBSTRING'LENGTH;
-------------------------------------------------------------------
begin
   SET(TO_THE_STRING,
        TO_THE_SIZE         => NEW_LENGTH,
        PRESERVE_THE_VALUE    => TRUE);
   TO_THE_STRING.THE_ITEMS((OLD_LENGTH + 1) .. NEW_LENGTH)
     := THE_SUBSTRING;
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
     raise OVERFLOW;
-------------------------------------------------------------------
end APPEND;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure APPEND(
-------------------------------------------------------------------
THE_ITEM       : in        ITEM;
TO_THE_STRING : in out  STRING) is
-------------------------------------------------------------------
   NEW_LENGTH :  NATURAL :=
                 TO_THE_STRING.THE_LENGTH + 1;
-------------------------------------------------------------------
begin
   SET(TO_THE_STRING,
        TO_THE_SIZE         => NEW_LENGTH,
        PRESERVE_THE_VALUE    => TRUE);
   TO_THE_STRING.THE_ITEMS(NEW_LENGTH) := THE_ITEM;
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
     raise OVERFLOW;
-------------------------------------------------------------------
end APPEND;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure INSERT(
-------------------------------------------------------------------
THE_STRING          : in      STRING;
IN_THE_STRING       : in out  STRING;
AT_THE_POSITION     : in        POSITIVE) is
-------------------------------------------------------------------
   OLD_LENGTH     : NATURAL :=  IN_THE_STRING.THE_LENGTH;
   NEW_LENGTH     : NATURAL :=  IN_THE_STRING.THE_LENGTH +
```

```
                                THE_STRING.THE_LENGTH;
   END_POSITION   : NATURAL :=
                    AT_THE_POSITION + THE_STRING.THE_LENGTH;
-------------------------------------------------------------------
begin
   if AT_THE_POSITION > IN_THE_STRING.THE_LENGTH then
     raise POSITION_ERROR;
   else
     SET(IN_THE_STRING,
         TO_THE_SIZE        => NEW_LENGTH,
         PRESERVE_THE_VALUE   => TRUE);
     IN_THE_STRING.THE_ITEMS(END_POSITION .. NEW_LENGTH) :=
       IN_THE_STRING.THE_ITEMS(AT_THE_POSITION .. OLD_LENGTH);
     IN_THE_STRING.THE_ITEMS(AT_THE_POSITION ..
                             (END_POSITION - 1)) :=
     THE_STRING.THE_ITEMS(1 .. THE_STRING.THE_LENGTH);
   end if;
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
     raise OVERFLOW;
-------------------------------------------------------------------
end INSERT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure INSERT(
-------------------------------------------------------------------
THE_SUBSTRING    : in     SUBSTRING;
IN_THE_STRING    : in out STRING;
AT_THE_POSITION  : in     POSITIVE) is
-------------------------------------------------------------------
   OLD_LENGTH     : NATURAL := IN_THE_STRING.THE_LENGTH;
   NEW_LENGTH     : NATURAL :=
                    IN_THE_STRING.THE_LENGTH +
                    THE_SUBSTRING'LENGTH;
   END_POSITION   : NATURAL :=
                    AT_THE_POSITION + THE_SUBSTRING'LENGTH;
-------------------------------------------------------------------
begin
   if AT_THE_POSITION > IN_THE_STRING.THE_LENGTH then
     raise POSITION_ERROR;
   else
     SET(IN_THE_STRING,
         TO_THE_SIZE        => NEW_LENGTH,
         PRESERVE_THE_VALUE   => TRUE);
     IN_THE_STRING.THE_ITEMS(END_POSITION .. NEW_LENGTH) :=
       IN_THE_STRING.THE_ITEMS(AT_THE_POSITION .. OLD_LENGTH);
     IN_THE_STRING.THE_ITEMS(AT_THE_POSITION ..
                             (END_POSITION - 1)) :=
     THE_SUBSTRING;
   end if;
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
     raise OVERFLOW;
-------------------------------------------------------------------
end INSERT;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure DELETE(
-------------------------------------------------------------------
IN_THE_STRING      : in out  STRING;
FROM_THE_POSITION  : in      POSITIVE;
TO_THE_POSITION    : in      POSITIVE) is
-------------------------------------------------------------------
  NEW_LENGTH : NATURAL;
-------------------------------------------------------------------
begin
  if (FROM_THE_POSITION > IN_THE_STRING.THE_LENGTH)   or else
    (TO_THE_POSITION > IN_THE_STRING.THE_LENGTH)      or else
    (FROM_THE_POSITION > TO_THE_POSITION) then
    raise POSITION_ERROR;
  else
    NEW_LENGTH := IN_THE_STRING.THE_LENGTH -
              (TO_THE_POSITION - FROM_THE_POSITION + 1);
    IN_THE_STRING.THE_ITEMS(FROM_THE_POSITION .. NEW_LENGTH) :=
      IN_THE_STRING.THE_ITEMS
        ((TO_THE_POSITION + 1) .. IN_THE_STRING.THE_LENGTH);
    SET(   IN_THE_STRING,
         TO_THE_SIZE         => NEW_LENGTH,
         PRESERVE_THE_VALUE  => TRUE);
  end if;
-------------------------------------------------------------------
end DELETE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure REPLACE(
-------------------------------------------------------------------
IN_THE_STRING      : in out  STRING;
AT_THE_POSITION    : in      POSITIVE;
WITH_THE_STRING    : in      STRING) is
-------------------------------------------------------------------
  END_POSITION : NATURAL :=   AT_THE_POSITION +
                              WITH_THE_STRING.THE_LENGTH - 1;
-------------------------------------------------------------------
begin
  if (AT_THE_POSITION > IN_THE_STRING.THE_LENGTH) or else
    (END_POSITION > IN_THE_STRING.THE_LENGTH) then
    raise POSITION_ERROR;
  else
    IN_THE_STRING.THE_ITEMS(AT_THE_POSITION .. END_POSITION) :=
    WITH_THE_STRING.THE_ITEMS(1 .. WITH_THE_STRING.THE_LENGTH);
  end if;
-------------------------------------------------------------------
end REPLACE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure REPLACE(
-------------------------------------------------------------------
IN_THE_STRING        : in out  STRING;
AT_THE_POSITION      : in      POSITIVE;
WITH_THE_SUBSTRING   : in      SUBSTRING) is
-------------------------------------------------------------------
  END_POSITION :  NATURAL :=
```

```
                    AT_THE_POSITION + WITH_THE_SUBSTRING'
                                        LENGTH - 1;
-------------------------------------------------------------------
begin
   if (AT_THE_POSITION > IN_THE_STRING.THE_LENGTH) or else
      (END_POSITION > IN_THE_STRING.THE_LENGTH) then
      raise POSITION_ERROR;
   else
      IN_THE_STRING.THE_ITEMS(AT_THE_POSITION .. END_POSITION) :=
        WITH_THE_SUBSTRING;
   end if;
-------------------------------------------------------------------
end REPLACE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_ITEM(
-------------------------------------------------------------------
IN_THE_STRING    : in out  STRING;
AT_THE_POSITION  : in        POSITIVE;
WITH_THE_ITEM    : in        ITEM) is
-------------------------------------------------------------------
begin
   if AT_THE_POSITION > IN_THE_STRING.THE_LENGTH then
      raise POSITION_ERROR;
   else
      IN_THE_STRING.THE_ITEMS(AT_THE_POSITION) := WITH_THE_ITEM;
   end if;
-------------------------------------------------------------------
end SET_ITEM;
-------------------------------------------------------------------


-------------------------------------------------------------------
function COPY(
-------------------------------------------------------------------
FROM_THE_STRING : in STRING) return STRING is
-------------------------------------------------------------------
   TO_THE_STRING : STRING;
-------------------------------------------------------------------
begin
   SET(TO_THE_STRING,
       TO_THE_SIZE           => FROM_THE_STRING.THE_LENGTH,
       PRESERVE_THE_VALUE    => FALSE);
   TO_THE_STRING.THE_ITEMS(1..FROM_THE_STRING.THE_LENGTH) :=
     FROM_THE_STRING.THE_ITEMS(1..FROM_THE_STRING.THE_LENGTH);
   return TO_THE_STRING;
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
      raise OVERFLOW;
-------------------------------------------------------------------
end COPY;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_EQUAL(
```

```
-----------------------------------------------------------------
LEFT   : in STRING;
RIGHT  : in STRING) return BOOLEAN is
-----------------------------------------------------------------
begin
   if LEFT.THE_LENGTH /= RIGHT.THE_LENGTH then
      return FALSE;
   else
      for INDEX in 1 .. LEFT.THE_LENGTH
      loop
         if LEFT.THE_ITEMS(INDEX) /= RIGHT.THE_ITEMS(INDEX) then
            return FALSE;
         end if;
      end loop;
      return TRUE;
   end if;
-----------------------------------------------------------------
end IS_EQUAL;
-----------------------------------------------------------------


-----------------------------------------------------------------
function IS_EQUAL(
-----------------------------------------------------------------
LEFT   : in SUBSTRING;
RIGHT  : in STRING) return BOOLEAN is
-----------------------------------------------------------------
begin
   if LEFT'LENGTH /= RIGHT.THE_LENGTH then
      return FALSE;
   else
      for INDEX in 1 .. LEFT'LENGTH
      loop
         if   LEFT(LEFT'FIRST + INDEX - 1) /=
              RIGHT.THE_ITEMS(INDEX) then
            return FALSE;
         end if;
      end loop;
      return TRUE;
   end if;
-----------------------------------------------------------------
end IS_EQUAL;
-----------------------------------------------------------------


-----------------------------------------------------------------
function IS_EQUAL(
-----------------------------------------------------------------
LEFT   : in STRING;
RIGHT  : in SUBSTRING)  return BOOLEAN is
-----------------------------------------------------------------
begin
   if LEFT.THE_LENGTH /= RIGHT'LENGTH then
      return FALSE;
   else
      for INDEX in 1 .. LEFT.THE_LENGTH
      loop
         if   LEFT.THE_ITEMS(INDEX) /=
              RIGHT(RIGHT'FIRST + INDEX - 1) then
            return FALSE;
```

```
      end if;
    end loop;
    return TRUE;
  end if;
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_LESS_THAN(
-------------------------------------------------------------------
LEFT   : in STRING;
RIGHT  : in STRING) return BOOLEAN is
-------------------------------------------------------------------
begin
  for INDEX in 1 .. LEFT.THE_LENGTH
  loop
    if INDEX > RIGHT.THE_LENGTH then
      return FALSE;
    elsif LEFT.THE_ITEMS(INDEX) < RIGHT.THE_ITEMS(INDEX) then
      return TRUE;
    elsif RIGHT.THE_ITEMS(INDEX) < LEFT.THE_ITEMS(INDEX) then
      return FALSE;
    end if;
  end loop;
  return (LEFT.THE_LENGTH < RIGHT.THE_LENGTH);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_LESS_THAN(
-------------------------------------------------------------------
LEFT   : in SUBSTRING;
RIGHT  : in STRING) return BOOLEAN is
-------------------------------------------------------------------
begin
  for INDEX in 1 .. LEFT'LENGTH
  loop
    if INDEX > RIGHT.THE_LENGTH then
      return FALSE;
    elsif  LEFT(LEFT'FIRST + INDEX -1) <
           RIGHT.THE_ITEMS(INDEX) then
      return TRUE;
    elsif  RIGHT.THE_ITEMS(INDEX) <
           LEFT(LEFT'FIRST + INDEX -1) then
      return FALSE;
    end if;
  end loop;
  return (LEFT'LENGTH < RIGHT.THE_LENGTH);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_LESS_THAN(
```

```
-------------------------------------------------------------------
LEFT   : in STRING;
RIGHT  : in SUBSTRING) return BOOLEAN is
-------------------------------------------------------------------
begin
   for INDEX in 1 .. LEFT.THE_LENGTH
   loop
      if INDEX > RIGHT'LENGTH then
         return FALSE;
      elsif  LEFT.THE_ITEMS(INDEX) <
             RIGHT(RIGHT'FIRST + INDEX -1) then
         return TRUE;
      elsif RIGHT(RIGHT'FIRST + INDEX -1) <
          LEFT.THE_ITEMS(INDEX) then
         return FALSE;
      end if;
   end loop;
   return (LEFT.THE_LENGTH < RIGHT'LENGTH);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_GREATER_THAN(
-------------------------------------------------------------------
LEFT   : in STRING;
RIGHT  : in STRING) return BOOLEAN is
-------------------------------------------------------------------
begin
   for INDEX in 1 .. LEFT.THE_LENGTH
   loop
      if INDEX > RIGHT.THE_LENGTH then
         return TRUE;
      elsif LEFT.THE_ITEMS(INDEX) < RIGHT.THE_ITEMS(INDEX) then
         return FALSE;
      elsif RIGHT.THE_ITEMS(INDEX) < LEFT.THE_ITEMS(INDEX) then
         return TRUE;
      end if;
   end loop;
   return FALSE;
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_GREATER_THAN(
-------------------------------------------------------------------
LEFT   : in SUBSTRING;
RIGHT  : in STRING)  return BOOLEAN is
-------------------------------------------------------------------
begin
   for INDEX in 1 .. LEFT'LENGTH
   loop
      if INDEX > RIGHT.THE_LENGTH then
         return TRUE;
      elsif  LEFT(LEFT'FIRST + INDEX -1) <
             RIGHT.THE_ITEMS(INDEX) then
         return FALSE;
```

```
      elsif  RIGHT.THE_ITEMS(INDEX) <
             LEFT(LEFT'FIRST + INDEX -1) then
        return TRUE;
      end if;
    end loop;
    return FALSE;
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_GREATER_THAN(
-------------------------------------------------------------------
LEFT   : in STRING;
RIGHT  : in SUBSTRING) return BOOLEAN is
-------------------------------------------------------------------
begin
   for INDEX in 1 .. LEFT.THE_LENGTH
   loop
     if INDEX > RIGHT'LENGTH then
        return TRUE;
     elsif  LEFT.THE_ITEMS(INDEX) <
            RIGHT(RIGHT'FIRST + INDEX -1) then
        return FALSE;
     elsif  RIGHT(RIGHT'FIRST + INDEX -1) <
            LEFT.THE_ITEMS(INDEX) then
        return TRUE;
     end if;
   end loop;
   return FALSE;
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function LENGTH_OF(
-------------------------------------------------------------------
THE_STRING  : in STRING) return NATURAL is
-------------------------------------------------------------------
begin
   return THE_STRING.THE_LENGTH;
-------------------------------------------------------------------
end LENGTH_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_NULL(
-------------------------------------------------------------------
THE_STRING  : in STRING) return BOOLEAN is
-------------------------------------------------------------------
begin
   return (THE_STRING.THE_LENGTH = 0);
-------------------------------------------------------------------
end IS_NULL;
-------------------------------------------------------------------
-------------------------------------------------------------------
function ITEM_OF(
```

```
   ------------------------------------------------------------------
   THE_STRING       : in STRING;
   AT_THE_POSITION  : in POSITIVE)   return ITEM is
   ------------------------------------------------------------------
   begin
      if AT_THE_POSITION > THE_STRING.THE_LENGTH then
         raise POSITION_ERROR;
      else
         return THE_STRING.THE_ITEMS(AT_THE_POSITION);
      end if;
   ------------------------------------------------------------------
   end ITEM_OF;
   ------------------------------------------------------------------


   ------------------------------------------------------------------
   function SUBSTRING_OF(
   ------------------------------------------------------------------
   THE_STRING  : in STRING) return SUBSTRING is
   ------------------------------------------------------------------
      TEMPORARY_STRUCTURE : SUBSTRING(1 .. 1);
   ------------------------------------------------------------------
   begin
      return THE_STRING.THE_ITEMS(1 .. THE_STRING.THE_LENGTH);
   ------------------------------------------------------------------
   exception
      when CONSTRAINT_ERROR =>
         return TEMPORARY_STRUCTURE(1 .. 0);
   ------------------------------------------------------------------
   end SUBSTRING_OF;
   ------------------------------------------------------------------


   ------------------------------------------------------------------
   function SUBSTRING_OF(
   ------------------------------------------------------------------
   THE_STRING          : in STRING;
   FROM_THE_POSITION   : in POSITIVE;
   TO_THE_POSITION     : in POSITIVE)
   return SUBSTRING is
   ------------------------------------------------------------------
   begin
      if (FROM_THE_POSITION > THE_STRING.THE_LENGTH) or else
         (TO_THE_POSITION > THE_STRING.THE_LENGTH) or else
         (FROM_THE_POSITION > TO_THE_POSITION) then
         raise POSITION_ERROR;
      else
         return THE_STRING.THE_ITEMS(FROM_THE_POSITION ..
                TO_THE_POSITION);
      end if;
   ------------------------------------------------------------------
   end SUBSTRING_OF;
   ------------------------------------------------------------------


------------------------------------------------------------------
end DYNAMIC_STRING_PACKAGE;
------------------------------------------------------------------
```

List

```
------------------------------------------------------------------------
--
-- Unit     : GENERIC_LIST_PACKAGE specification
-- Name     : A Harrison Software Engineering Group, Cranfield
--            University,
--            RMCS, Shrivenham
-- Date     : 20 Dec 1990
-- Version  : 1
-- Function : List specification
--
------------------------------------------------------------------------
generic

   type ITEM_TYPE is private;

   with function IS_EQUAL(
   LEFT   : in ITEM_TYPE;
   RIGHT  : in ITEM_TYPE) return BOOLEAN;

------------------------------------------------------------------------
package GENERIC_LIST_PACKAGE is
------------------------------------------------------------------------

   type LIST_TYPE is private;

   procedure PUT_ON_BACK_OF(
   LIST        : in out LIST_TYPE;
   THIS_ITEM   : in     ITEM_TYPE);

   procedure GET_FROM_BACK_OF(
   LIST        : in out LIST_TYPE;
   THIS_ITEM   :    out ITEM_TYPE);

   procedure DELETE_BACK_OF(
   LIST        : in out LIST_TYPE);

   procedure PUT_ON_FRONT_OF(
   LIST        : in out LIST_TYPE;
   THIS_ITEM   : in     ITEM_TYPE);

   procedure GET_FROM_FRONT_OF(
   LIST        : in out LIST_TYPE;
   THIS_ITEM   :    out ITEM_TYPE);

   procedure DELETE_FRONT_OF(
   LIST        : in out LIST_TYPE);

   procedure DELETE_FROM(
   LIST        : in out LIST_TYPE;
   THIS_ITEM   : in     ITEM_TYPE);

   procedure CLEAR(
   LIST        : in out LIST_TYPE);

   function FRONT_OF(
   LIST        : in   LIST_TYPE)  return ITEM_TYPE;
```

```
   function BACK_OF(
   LIST         : in   LIST_TYPE)  return ITEM_TYPE;

   function LENGTH_OF(
   LIST         : in   LIST_TYPE)     return NATURAL;

   function IS_EMPTY(
   LIST         : in   LIST_TYPE)     return BOOLEAN;

   function IS_IN(
   LIST         : in   LIST_TYPE;
   ITEM         : in   ITEM_TYPE)     return BOOLEAN;

   function POSITION_IN(
   LIST         : in   LIST_TYPE;
   ITEM         : in   ITEM_TYPE)     return NATURAL;

   function SUCCESSOR(
   LIST         : in   LIST_TYPE;
   THIS_ITEM    : in   ITEM_TYPE)     return ITEM_TYPE;

   function PREDECESSOR(
   LIST         : in   LIST_TYPE;
   THIS_ITEM    : in   ITEM_TYPE)     return ITEM_TYPE;

   function ITEM_AT(
   LIST           : in   LIST_TYPE;
   NODE_NUMBER    : in   POSITIVE)   return ITEM_TYPE;

   OVERFLOW     : exception;
   UNDERFLOW    : exception;
-------------------------------------------------------------------------
private

   type NODE_TYPE;
   type NODE_POINTER_TYPE is access NODE_TYPE;
   type LIST_TYPE is
   record
     FRONT    : NODE_POINTER_TYPE;
     BACK     : NODE_POINTER_TYPE;
   end record;

-------------------------------------------------------------------------
end GENERIC_LIST_PACKAGE;
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unit      : GENERIC_LIST_PACKAGE body
-- Name      : A Harrison Software Engineering Group,
--             Cranfield University,
--             RMCS, Shrivenham
-- Date      : 20 Dec 1990
-- Version   : 1
-- Function : A Generic LIST body
--
------------------------------------------------------------------
with   TEXT_IO, UNCHECKED_DEALLOCATION;
use    TEXT_IO;
------------------------------------------------------------------
package body GENERIC_LIST_PACKAGE is
------------------------------------------------------------------

  type NODE_TYPE is
  record
    FORWARD     : NODE_POINTER_TYPE;
    BACKWARD    : NODE_POINTER_TYPE;
    THE_ITEM    : ITEM_TYPE;
  end record;

  ------------------------------------------------------------------
  procedure DISPOSE is new  UNCHECKED_DEALLOCATION(
                            NODE_TYPE,
                            NODE_POINTER_TYPE);
  ------------------------------------------------------------------

  ------------------------------------------------------------------
  procedure PUT_ON_BACK_OF(
  ------------------------------------------------------------------
  LIST        : in out  LIST_TYPE;
  THIS_ITEM   : in      ITEM_TYPE) is
  ------------------------------------------------------------------
  begin
    LIST.BACK := new NODE_TYPE'( LIST.BACK, null, THIS_ITEM);

    if LIST.BACK.FORWARD = null then
      LIST.FRONT := LIST.BACK;
    else
      LIST.BACK.FORWARD.BACKWARD := LIST.BACK;
    end if;
  ------------------------------------------------------------------
  exception
    when STORAGE_ERROR =>
      PUT_LINE
      ("Heap exhausted in GENERIC_LIST_PACKAGE at PUT_ON_BACK_OF");
      NEW_LINE;
  ------------------------------------------------------------------
  end PUT_ON_BACK_OF;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure GET_FROM_BACK_OF(
  ------------------------------------------------------------------
  LIST        : in out LIST_TYPE;
  THIS_ITEM   :    out ITEM_TYPE) is
```

```
   ------------------------------------------------------------------
   begin
      if LIST.BACK = null then
         raise UNDERFLOW;
      else
         THIS_ITEM := LIST.BACK.THE_ITEM;
         DELETE_BACK_OF(LIST);
      end if;
   ------------------------------------------------------------------
   exception
     when UNDERFLOW =>
         PUT_LINE
         ("Attemp to GET_FROM_BACK_OF a list in
GENERIC_LIST_PACKAGE");
         NEW_LINE;
   ------------------------------------------------------------------
   end GET_FROM_BACK_OF;
   ------------------------------------------------------------------


   ------------------------------------------------------------------
   procedure DELETE_BACK_OF(
   ------------------------------------------------------------------
   LIST : in out LIST_TYPE) is
   ------------------------------------------------------------------
      GARBAGE : NODE_POINTER_TYPE;
   ------------------------------------------------------------------
   begin
      if IS_EMPTY(LIST) then
         raise UNDERFLOW;
      else
         GARBAGE := LIST.BACK;
         if LIST.FRONT = LIST.BACK then      --.only one item in list
            LIST.FRONT := null;
            LIST.BACK := null;
         else                        -- more than one item in list
            LIST.BACK := GARBAGE.FORWARD;
            LIST.BACK.BACKWARD := null;
         end if;

         -- clear up

         GARBAGE.FORWARD := null;
         DISPOSE(GARBAGE);
      end if;
   ------------------------------------------------------------------
   exception
        when UNDERFLOW =>
           TEXT_IO.
            PUT_LINE("Attempt to DELETE_BACK_OF an empty list " &
                     "in LIST_PACKAGE");
           NEW_LINE;
   ------------------------------------------------------------------
   end DELETE_BACK_OF;
   ------------------------------------------------------------------


   ------------------------------------------------------------------
   procedure PUT_ON_FRONT_OF(
   ------------------------------------------------------------------
```

```
LIST        : in out    LIST_TYPE;
THIS_ITEM   : in      ITEM_TYPE) is
-------------------------------------------------------------------
begin
   LIST.FRONT := NEW NODE_TYPE'(null, LIST.FRONT, THIS_ITEM);
   if LIST.FRONT.BACKWARD = null then
      LIST.BACK := LIST.FRONT;
   else
      LIST.FRONT.BACKWARD.FORWARD := LIST.FRONT;
   end if;
-------------------------------------------------------------------
exception
   when STORAGE_ERROR =>
      PUT_LINE
      ("No memory for allocation in GENERIC_LIST_PACKAGE " &
       "at PUT_ON_FRONT_OF");
      NEW_LINE;
-------------------------------------------------------------------
end PUT_ON_FRONT_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET_FROM_FRONT_OF(
-------------------------------------------------------------------
LIST        : in out LIST_TYPE;
THIS_ITEM   :    out ITEM_TYPE) is
-------------------------------------------------------------------
begin
   if LIST.FRONT = null then
      raise UNDERFLOW;
   else
      THIS_ITEM := LIST.FRONT.THE_ITEM;
      DELETE_FRONT_OF(LIST);
   end if;
-------------------------------------------------------------------
exception
   when UNDERFLOW =>
      PUT_LINE("Attempt to GET_FROM_FRONT_OF empty list " &
               "in GENERIC_LIST_PACKAGE");
      NEW_LINE;
-------------------------------------------------------------------
end GET_FROM_FRONT_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure DELETE_FRONT_OF(
-------------------------------------------------------------------
LIST   : in out LIST_TYPE) is
-------------------------------------------------------------------
   GARBAGE : NODE_POINTER_TYPE;
-------------------------------------------------------------------
begin
   if LIST.FRONT = null then
      raise UNDERFLOW;
   else
      GARBAGE := LIST.FRONT;
      if LIST.FRONT = LIST.BACK then -- only one
         LIST.FRONT := null;
```

```
      LIST.BACK := null;
    else -- more than one
      LIST.FRONT := LIST.FRONT.BACKWARD;
      LIST.FRONT.FORWARD := null;
    end if;
  end if;

  -- clean up

  DISPOSE(GARBAGE);
-------------------------------------------------------------------
exception
  when UNDERFLOW =>
    PUT_LINE("Attempt to delete an item at DELETE_FRONT_OF " &
             "in GENERIC_LIST_PACKAGE");
    NEW_LINE;
-------------------------------------------------------------------
end DELETE_FRONT_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure DELETE_FROM(
-------------------------------------------------------------------
LIST       : in out LIST_TYPE;
THIS_ITEM  : in     ITEM_TYPE) is
-------------------------------------------------------------------
  NODE_POINTER : NODE_POINTER_TYPE;
-------------------------------------------------------------------
begin
  if LIST.FRONT = null then
    raise UNDERFLOW;
  else -- find item
    NODE_POINTER := LIST.FRONT;
    while NODE_POINTER /= null
    loop
      if IS_EQUAL(NODE_POINTER.THE_ITEM, THIS_ITEM) then
        exit;
      end if;
      NODE_POINTER := NODE_POINTER.BACKWARD;
    end loop;
    if NODE_POINTER /= null then -- delete it
      if NODE_POINTER = LIST.FRONT then
        DELETE_FRONT_OF(LIST);
      elsif NODE_POINTER = LIST.BACK then
        DELETE_BACK_OF(LIST);
      else -- item between ends
        NODE_POINTER.BACKWARD.FORWARD := NODE_POINTER.FORWARD;
        NODE_POINTER.FORWARD.BACKWARD := NODE_POINTER.BACKWARD;
      end if;
    end if;

    -- clean up

    if NODE_POINTER /= null then
      DISPOSE(NODE_POINTER);
    end if;
  end if;
-------------------------------------------------------------------
exception
```

```
   when UNDERFLOW =>
      PUT_LINE("Attempt to DELETE_FROM an empty list " &
               "in LIST_PACKAGE");
      NEW_LINE;
-------------------------------------------------------------------
end DELETE_FROM;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure CLEAR(
-------------------------------------------------------------------
LIST : in out LIST_TYPE) is
-------------------------------------------------------------------
begin

   -- Need to collect garbage here!!!!

   LIST.FRONT := null;
   LIST.BACK := null;
-------------------------------------------------------------------
end CLEAR;
-------------------------------------------------------------------

-------------------------------------------------------------------
function FRONT_OF(
-------------------------------------------------------------------
LIST : in LIST_TYPE) return ITEM_TYPE is
-------------------------------------------------------------------
begin
   if LIST.FRONT = null then
      raise UNDERFLOW;
   else
      return LIST.FRONT.THE_ITEM;
   end if;
-------------------------------------------------------------------
exception
   when UNDERFLOW =>
      PUT_LINE("Attempt to probe FRONT_OF empty list " &
               "in GENERIC_LIST_PACKAGE");
      NEW_LINE;

      -- Note system will crash from here since no valid value to
      -- return!!
-------------------------------------------------------------------
end FRONT_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
function BACK_OF(
-------------------------------------------------------------------
LIST : in LIST_TYPE) return ITEM_TYPE is
-------------------------------------------------------------------
begin
   if LIST.BACK = null then
      raise UNDERFLOW;
   else
      return LIST.BACK.THE_ITEM;
```

```
    end if;
  ------------------------------------------------------------------
  exception
    when UNDERFLOW =>
      PUT_LINE
      ("Probe BACK_OF empty list in GENERIC_LIST_PACKAGE");
      NEW_LINE;

      -- Note system will crash from here since no valid value to
      -- return
  ------------------------------------------------------------------
  end BACK_OF;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function LENGTH_OF(
  ------------------------------------------------------------------
  LIST : in LIST_TYPE) return NATURAL is
  ------------------------------------------------------------------
    LENGTH : NATURAL := 0;
    NODE_POINTER : NODE_POINTER_TYPE := LIST.FRONT;
  ------------------------------------------------------------------
  begin
    while NODE_POINTER /= null
    loop
      LENGTH := LENGTH + 1;
      NODE_POINTER := NODE_POINTER.BACKWARD;
    end loop;

    return LENGTH;
  ------------------------------------------------------------------
  end LENGTH_OF;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function IS_EMPTY(
  ------------------------------------------------------------------
  LIST : in LIST_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
  begin
    return (LIST.FRONT = null) and (LIST.FRONT = LIST.BACK);
  ------------------------------------------------------------------
  end IS_EMPTY;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function IS_IN(
  ------------------------------------------------------------------
  LIST : in LIST_TYPE;
  ITEM : in ITEM_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
    NODE_POINTER : NODE_POINTER_TYPE := LIST.FRONT;
  ------------------------------------------------------------------
  begin
    while NODE_POINTER /= null
    loop
      if IS_EQUAL(NODE_POINTER.THE_ITEM, ITEM) then
```

```
        return TRUE;
      end if;
      NODE_POINTER := NODE_POINTER.BACKWARD;
    end loop;
    return FALSE;
-------------------------------------------------------------------
end IS_IN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function POSITION_IN(
-------------------------------------------------------------------
LIST : in LIST_TYPE;
ITEM : in ITEM_TYPE) return NATURAL is
-------------------------------------------------------------------
    NODE_POINTER  : NODE_POINTER_TYPE := LIST.FRONT;
    INDEX         : NATURAL := 0;
-------------------------------------------------------------------
begin
    while NODE_POINTER /= null
    loop
      INDEX := INDEX + 1;
      if IS_EQUAL(NODE_POINTER.THE_ITEM, ITEM) then
        return INDEX;
      end if;
      NODE_POINTER := NODE_POINTER.BACKWARD;
    end loop;
    return 0;
-------------------------------------------------------------------
end POSITION_IN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function SUCCESSOR(
-------------------------------------------------------------------
LIST        : in LIST_TYPE;
THIS_ITEM   : in ITEM_TYPE) return ITEM_TYPE is
-------------------------------------------------------------------
    NODE_POINTER : NODE_POINTER_TYPE := LIST.FRONT;
-------------------------------------------------------------------
begin
    while NODE_POINTER /= null
    loop
      if IS_EQUAL(NODE_POINTER.THE_ITEM, THIS_ITEM) then
        if NODE_POINTER.BACKWARD /= null then
          return NODE_POINTER.BACKWARD.THE_ITEM;
        else
          raise OVERFLOW;
        end if;
      else
        NODE_POINTER := NODE_POINTER.BACKWARD;
      end if;
    end loop;

    raise OVERFLOW;
-------------------------------------------------------------------
exception
    when OVERFLOW =>
```

```
      PUT_LINE("This item has no successor");
      return THIS_ITEM;
-------------------------------------------------------------------
end SUCCESSOR;
-------------------------------------------------------------------


-------------------------------------------------------------------
function PREDECESSOR(
-------------------------------------------------------------------
LIST        : in LIST_TYPE;
THIS_ITEM   : in ITEM_TYPE) return ITEM_TYPE is
-------------------------------------------------------------------
  NODE_POINTER : NODE_POINTER_TYPE := LIST.BACK;
-------------------------------------------------------------------
begin
  while NODE_POINTER /= null
  loop
    if IS_EQUAL(NODE_POINTER.THE_ITEM, THIS_ITEM) then
      if NODE_POINTER.FORWARD /= null then
        return NODE_POINTER.FORWARD.THE_ITEM;
      else
        raise UNDERFLOW;
      end if;
    else
      NODE_POINTER := NODE_POINTER.FORWARD;
    end if;
  end loop;

  raise UNDERFLOW;
-------------------------------------------------------------------
exception
  when UNDERFLOW =>
    PUT_LINE("This item has no predecessor");
    return THIS_ITEM;
-------------------------------------------------------------------
end PREDECESSOR;
-------------------------------------------------------------------


-------------------------------------------------------------------
function ITEM_AT(
-------------------------------------------------------------------
LIST          : in LIST_TYPE;
NODE_NUMBER : in POSITIVE) return ITEM_TYPE is
-------------------------------------------------------------------
  COUNT          : POSITIVE := 1;
  NODE_POINTER   : NODE_POINTER_TYPE := LIST.FRONT;
-------------------------------------------------------------------
begin
  -- must be checked for empty and short list before entry
  if LIST.FRONT = null then
    raise UNDERFLOW;
  elsif LENGTH_OF(LIST) < NODE_NUMBER then
    raise OVERFLOW;
  else
    while COUNT /= NODE_NUMBER
    loop
      NODE_POINTER := NODE_POINTER.BACKWARD;
      COUNT := COUNT + 1;
```

```
      end loop;
      return NODE_POINTER.THE_ITEM;
    end if;
  ------------------------------------------------------------------
  exception
    when UNDERFLOW =>
      PUT_LINE("Empty list in ITEM_AT in LIST_PACKAGE");
    when OVERFLOW =>
      PUT_LINE("list too short in ITEM_AT in LIST_PACKAGE");
  ------------------------------------------------------------------
  end ITEM_AT;
  ------------------------------------------------------------------

------------------------------------------------------------------------
end GENERIC_LIST_PACKAGE;
------------------------------------------------------------------------
```

Tree

```
-------------------------------------------------------------------
--
-- Unit          : GENERIC_TREE_PACKAGE specification
-- Author        : A Harrison Software Engineering Group,
--                 Cranfield University,
--                 RMCS, Shrivenham
-- Date          : 14 December 1991
-- Function      : Binary search tree specification
--                 Algorithms & Data Structures Wirth 1986
-------------------------------------------------------------------
generic
  type ITEM_BASE_TYPE is private;
  type ITEM_COMPOSITE_TYPE is private;
  type ITEM_PTR_TYPE is access ITEM_COMPOSITE_TYPE;

  with function "<"(
  ITEM_PTR_1 : in ITEM_PTR_TYPE ;
  ITEM_PTR_2 : in ITEM_PTR_TYPE) return BOOLEAN;

  with function ">"(
  ITEM_PTR_1 : in ITEM_PTR_TYPE ;
  ITEM_PTR_2 : in ITEM_PTR_TYPE) return BOOLEAN;

  with function IS_EQUAL(
  ITEM     : in ITEM_BASE_TYPE;
  ITEM_PTR : in ITEM_PTR_TYPE) return BOOLEAN;

  with function IS_LESS_THAN(
  ITEM     : in ITEM_BASE_TYPE;
  ITEM_PTR : in ITEM_PTR_TYPE) return BOOLEAN;

  with function IS_GREATER_THAN(
  ITEM     : in ITEM_BASE_TYPE;
  ITEM_PTR : in ITEM_PTR_TYPE) return BOOLEAN;

  with procedure PUT(
  ITEM_PTR : in ITEM_PTR_TYPE);
-------------------------------------------------------------------
package GENERIC_TREE_PACKAGE is
-------------------------------------------------------------------

  type TREE_PTR_TYPE is private;
  type ITEM_ARRAY_TYPE is array(POSITIVE range <>) of ITEM_PTR_TYPE;

  procedure INSERT(
  ITEM_PTR : in ITEM_PTR_TYPE;
  TREE_PTR : in out TREE_PTR_TYPE);

  procedure DELETE
  (ITEM_PTR  : in ITEM_PTR_TYPE;
   TREE_PTR : in out TREE_PTR_TYPE);

  function IS_IN(
  ITEM     : in ITEM_BASE_TYPE;
  TREE_PTR : in TREE_PTR_TYPE) return BOOLEAN;

  procedure FIND(
```

```
  ITEM       : in     ITEM_BASE_TYPE;
  TREE_PTR   : in     TREE_PTR_TYPE;
  RESULT_PTR : in out ITEM_PTR_TYPE);

  procedure FORM_ARRAY(
  TREE_PTR    : in     TREE_PTR_TYPE;
  ITEM_ARRAY  :    out ITEM_ARRAY_TYPE;
  ITEM_NUMBER : in out NATURAL);

  procedure PRINT(
  TREE_PTR : in TREE_PTR_TYPE);

  function SIZE_OF(
  TREE_PTR : in TREE_PTR_TYPE) return NATURAL;

  procedure CLEAR(
  TREE_PTR : in out TREE_PTR_TYPE);

private

  type TREE_NODE_TYPE;
  type TREE_PTR_TYPE is access TREE_NODE_TYPE;

-------------------------------------------------------------------------
end GENERIC_TREE_PACKAGE;
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
--
-- Unit        : GENERIC_TREE_PACKAGE specification
-- Author      : A Harrison Software Engineering Group,
--               Cranfield University,
--               RMCS, Shrivenham
-- Date        : 14 December 1991
-- Function    : Binary search tree specification
--
------------------------------------------------------------------------
package body GENERIC_TREE_PACKAGE is
------------------------------------------------------------------------

  subtype BALANCE_RANGE is INTEGER range -1..1;

  type TREE_NODE_TYPE is
  record
    ITEM_PTR  : ITEM_PTR_TYPE;
    COUNT     : NATURAL := 0;
    LEFT_PTR  : TREE_PTR_TYPE;
    RIGHT_PTR : TREE_PTR_TYPE;
    BALANCE   : BALANCE_RANGE;
  end record;

  ----------------------------------------------------------------------
  procedure BALANCE_L(
  ----------------------------------------------------------------------
  TREE_PTR : in out TREE_PTR_TYPE;
  H        : in out BOOLEAN) is
  ----------------------------------------------------------------------
    T1, T2 : TREE_PTR_TYPE;
    B1, B2 : BALANCE_RANGE;
  ----------------------------------------------------------------------
  begin -- Left branch has shrunk
    case TREE_PTR.BALANCE is
      when -1 => TREE_PTR.BALANCE := 0;
      when  0 => TREE_PTR.BALANCE := 1;
                 H := FALSE;
      when  1 => T1 := TREE_PTR.RIGHT_PTR;  -- Rebalance
                 B1 := T1.BALANCE;
                 if B1 >= 0 then -- Single RR rotation
                   TREE_PTR.RIGHT_PTR := T1.LEFT_PTR;
                   T1.LEFT_PTR := TREE_PTR;
                   if B1 = 0 then
                     TREE_PTR.BALANCE := 1;
                     T1.BALANCE := -1;
                     H := FALSE;
                   else
                     TREE_PTR.BALANCE := 0;
                     T1.BALANCE := 0;
                   end if;
                   TREE_PTR := T1;
                 else  -- Double RL rotation
                   T2 := T1.LEFT_PTR;
                   B2 := T2.BALANCE;
                   T1.LEFT_PTR := T2.RIGHT_PTR;
                   T2.RIGHT_PTR := T1;
                   TREE_PTR.RIGHT_PTR := T2.LEFT_PTR;
                   T2.LEFT_PTR := TREE_PTR;
                   if B2 = 1 then
```

```
                    TREE_PTR.BALANCE := -1;
                  else
                    TREE_PTR.BALANCE := 0;
                  end if;
                  if B2 = -1 then
                    T1.BALANCE := 1;
                  else
                    T1.BALANCE := 0;
                  end if;
                  TREE_PTR := T2;
                  T2.BALANCE := 0;
                end if;
  end case;
-------------------------------------------------------------------
end BALANCE_L;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure BALANCE_R(
-------------------------------------------------------------------
TREE_PTR : in out TREE_PTR_TYPE;
H        : in out BOOLEAN) is
-------------------------------------------------------------------
  T1, T2 : TREE_PTR_TYPE;
  B1, B2 : BALANCE_RANGE;
-------------------------------------------------------------------
begin  -- Right branch has shrunk
  case  TREE_PTR.BALANCE is
    when  1 => TREE_PTR.BALANCE := 0;
    when  0 => TREE_PTR.BALANCE := -1;
               H := FALSE;
    when -1 => T1 := TREE_PTR.LEFT_PTR;  -- Rebalance
               B1 := T1.BALANCE;
               if B1 <= 0 then -- Single LL rotation
                 TREE_PTR.LEFT_PTR := T1.RIGHT_PTR;
                 T1.RIGHT_PTR := TREE_PTR;
                 if B1 = 0 then
                   TREE_PTR.BALANCE := -1;
                   T1.BALANCE := 1;
                   H := FALSE;
                 else
                   TREE_PTR.BALANCE := 0;
                   T1.BALANCE := 0;
                 end if;
                 TREE_PTR := T1;
               else -- Double LR rotation
                 T2 := T1.RIGHT_PTR;
                 B2 := T2.BALANCE;
                 T1.RIGHT_PTR := T2.LEFT_PTR;
                 T2.LEFT_PTR := T1;
                 TREE_PTR.LEFT_PTR := T2.RIGHT_PTR;
                 T2.RIGHT_PTR := TREE_PTR;
                 if B2 = -1 then
                   TREE_PTR.BALANCE := 1;
                 else
                   TREE_PTR.BALANCE := 0;
                 end if;
                 if B2 = 1 then
                   T1.BALANCE := -1;
```

```
                else
                  T1.BALANCE := 0;
                end if;
                TREE_PTR := T2;
                T2.BALANCE := 0;
              end if;
  end case;
-------------------------------------------------------------------
end BALANCE_R;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure DELETE(
-------------------------------------------------------------------
ITEM_PTR  : in     ITEM_PTR_TYPE;
TREE_PTR  : in out TREE_PTR_TYPE) is
H : BOOLEAN := FALSE;
-------------------------------------------------------------------
  procedure DELETE_ITEM(
  -----------------------------------------------------------------
  ITEM_PTR  : in ITEM_PTR_TYPE;
  TREE_PTR  : in out TREE_PTR_TYPE;
  H         : in out BOOLEAN) is
  -----------------------------------------------------------------
    TEMP_PTR : TREE_PTR_TYPE;
    -----------------------------------------------------------------
    procedure DEL(
    -----------------------------------------------------------------
    T : in out TREE_PTR_TYPE;
    H : in out BOOLEAN) is
    -----------------------------------------------------------------
    begin
      if T.RIGHT_PTR /= null then
        DEL(T.RIGHT_PTR, H);
        if H then
          BALANCE_R(T, H);
        end if;
      else
        TEMP_PTR.ITEM_PTR := T.ITEM_PTR;
        TEMP_PTR.COUNT := T.COUNT;
        TEMP_PTR := T;
        T := T.LEFT_PTR;
        H := TRUE;
      end if;
  -----------------------------------------------------------------
    end DEL;
  -----------------------------------------------------------------
  begin
    if TREE_PTR = null then
      null;
    elsif TREE_PTR.ITEM_PTR > ITEM_PTR then
      DELETE_ITEM(ITEM_PTR, TREE_PTR.LEFT_PTR, H);
      if H then
        BALANCE_L(TREE_PTR, H);
      end if;
    elsif TREE_PTR.ITEM_PTR < ITEM_PTR then
      DELETE_ITEM(ITEM_PTR, TREE_PTR.RIGHT_PTR, H);
      if H then
        BALANCE_R(TREE_PTR, H);
```

```
      end if;
    else
      TEMP_PTR := TREE_PTR;
      if TEMP_PTR.RIGHT_PTR = null then
        TREE_PTR := TEMP_PTR.LEFT_PTR;
        H:= TRUE;
      elsif TEMP_PTR.LEFT_PTR = null then
        TREE_PTR := TEMP_PTR.RIGHT_PTR;
        H := TRUE;
      else
        DEL(TEMP_PTR.LEFT_PTR, H);
        if H then
          BALANCE_L(TREE_PTR, H);
        end if;
      end if;
  end if;
  ------------------------------------------------------------------
  end DELETE_ITEM;
  ------------------------------------------------------------------

------------------------------------------------------------------
begin
  DELETE_ITEM
  (ITEM_PTR => ITEM_PTR,
   TREE_PTR => TREE_PTR,
    H       => H);
------------------------------------------------------------------
end DELETE;
------------------------------------------------------------------


------------------------------------------------------------------
procedure INSERT(
ITEM_PTR : in     ITEM_PTR_TYPE;
TREE_PTR : in out TREE_PTR_TYPE) is
------------------------------------------------------------------
  H : BOOLEAN := FALSE;
  ------------------------------------------------------------------
  procedure SEARCH(
  ------------------------------------------------------------------
  ITEM_PTR : in     ITEM_PTR_TYPE;
  TREE_PTR : in out TREE_PTR_TYPE;
  H        : in out BOOLEAN) is
  ------------------------------------------------------------------
    T1_PTR, T2_PTR : TREE_PTR_TYPE;
  ------------------------------------------------------------------
  begin

    if TREE_PTR = null then
       -- ITEM_PTR not in tree-insert it
       TREE_PTR := new TREE_NODE_TYPE'
                       (ITEM_PTR, 1, null, null, 0);
       H := TRUE;
   elsif ITEM_PTR < TREE_PTR.ITEM_PTR then
       SEARCH(ITEM_PTR, TREE_PTR.LEFT_PTR, H);
       if H then
          -- LEFT_PTR branch has grown higher
          case TREE_PTR.BALANCE is
             when 1   => TREE_PTR.BALANCE := 0;
                                 H := FALSE;
```

```
            when 0    => TREE_PTR.BALANCE := -1;
            when -1   => -- Rebalance
                         T1_PTR := TREE_PTR.LEFT_PTR;
                         if T1_PTR.BALANCE = -1 then
                           -- Single LL rotation
                           TREE_PTR.LEFT_PTR := T1_PTR.RIGHT_PTR;
                           T1_PTR.RIGHT_PTR := TREE_PTR;
                           TREE_PTR.BALANCE := 0;
                           TREE_PTR := T1_PTR;
                         else
                           -- Double LR rotation
                           T2_PTR := T1_PTR.RIGHT_PTR;
                           T1_PTR.RIGHT_PTR := T2_PTR.LEFT_PTR;
                           T2_PTR.LEFT_PTR := T1_PTR;
                           TREE_PTR.LEFT_PTR := T2_PTR.RIGHT_PTR;
                           T2_PTR.RIGHT_PTR := TREE_PTR;
                           if T2_PTR.BALANCE = -1 then
                             TREE_PTR.BALANCE := 1;
                           else
                             TREE_PTR.BALANCE := 0;
                           end if;
                           if T2_PTR.BALANCE = 1 then
                             T1_PTR.BALANCE := -1;
                           else
                             T1_PTR.BALANCE := 0;
                           end if;
                           TREE_PTR := T2_PTR;
                         end if;
                         TREE_PTR.BALANCE := 0;
                         H := FALSE;
        end case;
      end if;
    elsif ITEM_PTR > TREE_PTR.ITEM_PTR then
      SEARCH(ITEM_PTR, TREE_PTR.RIGHT_PTR, H);
           if H then
              case TREE_PTR.BALANCE is
              when -1   => TREE_PTR.BALANCE := 0;
                           H := FALSE;
              when 0    => TREE_PTR.BALANCE := 1;
              when 1    => -- Rebalance
                           T1_PTR := TREE_PTR.RIGHT_PTR;
                           if TREE_PTR.BALANCE = 1 then
                             -- Single RR rotation
                             TREE_PTR.RIGHT_PTR:=
                             T1_PTR.LEFT_PTR;
                             T1_PTR.LEFT_PTR := TREE_PTR;
                             TREE_PTR.BALANCE := 0;
                             TREE_PTR := T1_PTR;
                           else
                             -- Double RL rotation
                             T2_PTR := T1_PTR.LEFT_PTR;
                             T1_PTR.LEFT_PTR :=
                             T2_PTR.RIGHT_PTR;
                             T2_PTR.RIGHT_PTR := T1_PTR;
                             TREE_PTR.RIGHT_PTR :=
                             T2_PTR.LEFT_PTR;
                             T2_PTR.LEFT_PTR := TREE_PTR;
                             if T2_PTR.BALANCE = 1 then
                               TREE_PTR.BALANCE := -1;
                             else
```

```
                                    TREE_PTR.BALANCE := 0;
                                  end if;
                                  if T2_PTR.BALANCE = -1 then
                                    T1_PTR.BALANCE := 1;
                                  else
                                    T1_PTR.BALANCE := 0;
                                  end if;
                                  TREE_PTR := T2_PTR;
                                end if;
                                TREE_PTR.BALANCE := 0;
                                H := FALSE;
            end case;
        end if;
      else
        TREE_PTR.COUNT := TREE_PTR.COUNT + 1;
        --H := FALSE;
      end if;
  ------------------------------------------------------------------
  end SEARCH;
  ------------------------------------------------------------------

------------------------------------------------------------------
begin

   SEARCH(ITEM_PTR, TREE_PTR, H);

------------------------------------------------------------------
end INSERT;
------------------------------------------------------------------


------------------------------------------------------------------
function IS_IN(
------------------------------------------------------------------
ITEM     : in ITEM_BASE_TYPE;
TREE_PTR : in TREE_PTR_TYPE) return BOOLEAN is
------------------------------------------------------------------
begin

  if TREE_PTR = null then
    return FALSE;
  else
    if IS_EQUAL(ITEM, TREE_PTR.ITEM_PTR) then
      return TRUE;
    elsif IS_LESS_THAN(ITEM, TREE_PTR.ITEM_PTR) then
      return IS_IN(ITEM, TREE_PTR.LEFT_PTR);
    else
      return IS_IN(ITEM, TREE_PTR.RIGHT_PTR);
    end if;
  end if;
------------------------------------------------------------------
end IS_IN;
------------------------------------------------------------------


------------------------------------------------------------------
procedure FIND(
------------------------------------------------------------------
ITEM       : in     ITEM_BASE_TYPE;
TREE_PTR   : in     TREE_PTR_TYPE;
```

```
RESULT_PTR : in out ITEM_PTR_TYPE) is
-------------------------------------------------------------------
begin

  if TREE_PTR = null then
    return;
  else
    if IS_EQUAL(ITEM, TREE_PTR.ITEM_PTR) then
      RESULT_PTR := TREE_PTR.ITEM_PTR;
      return;
    elsif IS_LESS_THAN(ITEM, TREE_PTR.ITEM_PTR) then
      FIND(ITEM, TREE_PTR.LEFT_PTR, RESULT_PTR);
    else
      FIND(ITEM, TREE_PTR.RIGHT_PTR, RESULT_PTR);
    end if;
  end if;
-------------------------------------------------------------------
end FIND;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure FORM_ARRAY(
-------------------------------------------------------------------
TREE_PTR    : in     TREE_PTR_TYPE;
ITEM_ARRAY  :    out ITEM_ARRAY_TYPE;
ITEM_NUMBER : in out NATURAL) is
-------------------------------------------------------------------
begin

  if TREE_PTR = null then
    return;
  else

    ITEM_NUMBER := ITEM_NUMBER + 1;
    ITEM_ARRAY(ITEM_NUMBER) := TREE_PTR.ITEM_PTR;

    FORM_ARRAY
    (TREE_PTR    => TREE_PTR.LEFT_PTR,
     ITEM_ARRAY  => ITEM_ARRAY,
     ITEM_NUMBER => ITEM_NUMBER);

    FORM_ARRAY
    (TREE_PTR    => TREE_PTR.RIGHT_PTR,
     ITEM_ARRAY  => ITEM_ARRAY,
     ITEM_NUMBER => ITEM_NUMBER);

  end if;

-------------------------------------------------------------------
end FORM_ARRAY;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PRINT(
-------------------------------------------------------------------
TREE_PTR : in TREE_PTR_TYPE) is
-------------------------------------------------------------------
begin
```

```
    if TREE_PTR = null then
       return;
    else
      PRINT(TREE_PTR.LEFT_PTR);
      PUT(TREE_PTR.ITEM_PTR);
      PRINT(TREE_PTR.RIGHT_PTR);
  end if;
-------------------------------------------------------------------
end PRINT;
-------------------------------------------------------------------


-------------------------------------------------------------------
function SIZE_OF(
-------------------------------------------------------------------
TREE_PTR : in TREE_PTR_TYPE) return NATURAL is
-------------------------------------------------------------------
begin

   if TREE_PTR = null then
      return 0;
   else
     return SIZE_OF(TREE_PTR => TREE_PTR.LEFT_PTR)  +
            SIZE_OF(TREE_PTR => TREE_PTR.RIGHT_PTR) + 1;
  end if;
-------------------------------------------------------------------
end SIZE_OF;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure CLEAR(
-------------------------------------------------------------------
TREE_PTR : in out TREE_PTR_TYPE) is
-------------------------------------------------------------------
begin
  TREE_PTR := null;
-------------------------------------------------------------------
end CLEAR;
-------------------------------------------------------------------

-------------------------------------------------------------------
end GENERIC_TREE_PACKAGE;
-------------------------------------------------------------------
```

System Types

```
------------------------------------------------------------------
--
-- Unit      : SYSTEM_TYPES_PACKAGE Specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield, RMCS, Shrivenham
-- Date      : Dec 1990
-- Function  : This package provides an instantiation of the
--             dynamic string and other generally used types and
--             operations
--
------------------------------------------------------------------
with TEXT_IO, DYNAMIC_STRING_PACKAGE, GENERIC_LIST_PACKAGE;
------------------------------------------------------------------
package SYSTEM_TYPES_PACKAGE is
------------------------------------------------------------------
  package DYNAMIC_STRING is new DYNAMIC_STRING_PACKAGE(
  ----------------------------------------------------------------
  ITEM      => STANDARD.CHARACTER,
  SUBSTRING => STANDARD.STRING,
  "<"       => "<",
  PUT       => TEXT_IO.PUT);
  ----------------------------------------------------------------

  function CONVERT(
  DAY : in STANDARD.STRING) return STANDARD.STRING;

  procedure GET(
  TOKEN     : in out DYNAMIC_STRING.STRING;
  FROM_FILE : in     TEXT_IO.FILE_TYPE);

  procedure GET(
  FROM_FILE       : in out TEXT_IO.FILE_TYPE;
  TOKEN           :    out STANDARD.STRING;
  LENGTH_OF_TOKEN :    out NATURAL);

  function GET_FIELD(
  NUMBER  : in STANDARD.POSITIVE;
  FROM    : in STANDARD.STRING)
  return STANDARD.STRING;

  function GET_FIELDS(
  FROM_FIELD : in STANDARD.POSITIVE;
  TO_FIELD   : in STANDARD.POSITIVE;
  FROM       : in STANDARD.STRING)
  return STANDARD.STRING;

  function IS_FIELD_AT
  (POSITION : in STANDARD.POSITIVE;
   IN_STRING : in STANDARD.STRING)
  return BOOLEAN;

  ----------------------------------------------------------------
  package DYNAMIC_STRING_LIST_PACKAGE is new GENERIC_LIST_PACKAGE(
  ----------------------------------------------------------------
  ITEM_TYPE => DYNAMIC_STRING.
               STRING,
  IS_EQUAL  => DYNAMIC_STRING.
```

```
                IS_EQUAL);
  ------------------------------------------------------------------------

  type DYNAMIC_STRING_ARRAY is array(STANDARD.POSITIVE range <>)
                               of DYNAMIC_STRING.STRING;

------------------------------------------------------------------------
end SYSTEM_TYPES_PACKAGE;
------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unit      : SYSTEM_TYPES_PACKAGE Body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : Dec 1990
-- Function  : This package provides an instantiation of
--             the dynamic string and other generally used types and
--             operations
--
------------------------------------------------------------------
package body SYSTEM_TYPES_PACKAGE is
------------------------------------------------------------------

  ----------------------------------------------------------------
  function CONVERT(
  ----------------------------------------------------------------
  DAY : in STANDARD.STRING) return STANDARD.STRING is
  ----------------------------------------------------------------
    LOWER_CASE : STANDARD.STRING(1 .. DAY'LENGTH);
  ----------------------------------------------------------------
  begin

    -- Convert to lower case

    for CHAR in 1 .. DAY'LENGTH
    loop
      LOWER_CASE(CHAR) := CHARACTER'
                          VAL(CHARACTER'POS(DAY(CHAR)) + 32);
    end loop;

    return LOWER_CASE;
  ----------------------------------------------------------------
  exception

    WHEN OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_ESE_PACKAGE at " &
       "CONVERT");
  ----------------------------------------------------------------
    end CONVERT;
  ----------------------------------------------------------------


  ----------------------------------------------------------------
  procedure GET(
  ----------------------------------------------------------------
  TOKEN     : in out DYNAMIC_STRING.STRING;
  FROM_FILE : in     TEXT_IO.FILE_TYPE) is
  ----------------------------------------------------------------
    CHAR  : STANDARD.CHARACTER;
    SPACE : constant CHARACTER := ' ';
  ----------------------------------------------------------------
  begin
    TEXT_IO.GET(FROM_FILE, CHAR);

    -- Skip spaces

    while CHAR = SPACE
    loop
```

```
    TEXT_IO.GET(FROM_FILE, CHAR);
  end loop;

  -- Collect characters and append to a dynamic string

  loop
    SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.APPEND(CHAR, TOKEN);

    -- Stop if end of line has been reached

    exit when TEXT_IO.END_OF_LINE(FROM_FILE);
    TEXT_IO.GET(FROM_FILE, CHAR);

    -- Stop if end of word

    exit when CHAR = SPACE;
  end loop;
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET(
-------------------------------------------------------------------
FROM_FILE       : in out TEXT_IO.FILE_TYPE;
TOKEN           : out    STANDARD.STRING;
LENGTH_OF_TOKEN : out    NATURAL) is
-------------------------------------------------------------------
begin
  TEXT_IO.
  GET_LINE
  (FROM_FILE,
   TOKEN,
   LENGTH_OF_TOKEN);
  if not TEXT_IO.END_OF_FILE(FROM_FILE) and then
         TEXT_IO.END_OF_LINE(FROM_FILE) then
    TEXT_IO.SKIP_LINE(FROM_FILE);
  end if;

-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
function GET_FIELD(
-------------------------------------------------------------------
NUMBER : in STANDARD.POSITIVE;
FROM   : in STANDARD.STRING)
return STANDARD.STRING is
-------------------------------------------------------------------
  FIRST   : POSITIVE := 1;
  LAST    : POSITIVE := 1;
  COUNT   : NATURAL  := 0;
-------------------------------------------------------------------
begin
  if NUMBER = 1 then
    FIRST := FROM'FIRST;
  else
```

```
    -- Find position of first character in fild

    for INDEX in FROM'FIRST .. FROM'LAST
    loop
      if FROM(INDEX) = '_' then
        COUNT := COUNT + 1;
        if COUNT = NUMBER - 1 then
          FIRST := INDEX + 1;
          exit;
        end if;
      end if;
    end loop;
  end if;

  -- Find last character in field

  for INDEX in FIRST .. FROM'LENGTH
  loop
    if FROM(INDEX) = '_' then
      LAST := INDEX - 1;
      exit;
    end if;
  end loop;
  if LAST = 1 then -- Must be last field
    LAST := FROM'LAST;
  end if;
  return FROM(FIRST .. LAST);

------------------------------------------------------------------
exception

  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE(" CONSTRAINT_ERROR raised at GET_FIELD in " &
                     "SYSTEM_TYPES_PACKAGE");

------------------------------------------------------------------
end GET_FIELD;
------------------------------------------------------------------


------------------------------------------------------------------
function GET_FIELDS(
------------------------------------------------------------------
FROM_FIELD : in STANDARD.POSITIVE;
TO_FIELD   : in STANDARD.POSITIVE;
FROM       : in STANDARD.STRING)
return STANDARD.STRING is
------------------------------------------------------------------
  FIRST : POSITIVE := 1;
  LAST  : POSITIVE := 1;
  COUNT : NATURAL  := 0;
------------------------------------------------------------------
begin
  if FROM_FIELD = 1 then
    FIRST := FROM'FIRST;
  else

    -- Find position of first character in first field
```

```
    for INDEX in FROM'FIRST .. FROM'LAST
    loop
      if FROM(INDEX) = '_' then
        COUNT := COUNT + 1;
        if COUNT = FROM_FIELD - 1 then
          FIRST := INDEX + 1;
          exit;
        end if;
      end if;
    end loop;
  end if;

  -- Find position of last character of last field

  for INDEX in FIRST .. FROM'LAST
  loop
    if FROM(INDEX) = '_' then
      COUNT := COUNT + 1;
      if COUNT = TO_FIELD then
        LAST := INDEX - 1;
        exit;
      end if;
    end if;
  end loop;
  if LAST = 1 then -- Must be last field in string
    LAST := FROM'LAST;
  end if;
  return FROM(FIRST .. LAST);

------------------------------------------------------------------
exception

  when CONSTRAINT_ERROR =>
    TEXT_IO.
    PUT_LINE(" CONSTRAINT_ERROR raised at GET_FIELDS in " &
             "SYSTEM_TYPES_PACKAGE");

------------------------------------------------------------------
end GET_FIELDS;
------------------------------------------------------------------


------------------------------------------------------------------
function IS_FIELD_AT(
------------------------------------------------------------------
POSITION : in STANDARD.POSITIVE;
IN_STRING : in STANDARD.STRING)
return BOOLEAN is
------------------------------------------------------------------
  FIELD : STANDARD.NATURAL := 0;
  COUNT : STANDARD.NATURAL := 0;
------------------------------------------------------------------
begin
  if IN_STRING'LENGTH = 0 then
    return FALSE;  -- No fields
  else
    while COUNT < IN_STRING'LENGTH
    loop
      COUNT := COUNT + 1;
      if IN_STRING(COUNT) = '_' then
```

```
          FIELD := FIELD + 1;
          if FIELD = POSITION then
            return TRUE;   -- Deals with only one field and
                           -- cases where
          end if;          -- position is less than total
                           -- number of fields
        end if;
      end loop;
      -- Deal with the limiting cases
      if FIELD + 1 = POSITION then
        return TRUE;  -- Last field
      else
        return FALSE; -- Field beyond end of string
      end if;
    end if;
  ------------------------------------------------------------------
  end IS_FIELD_AT;
  ------------------------------------------------------------------


------------------------------------------------------------------
end SYSTEM_TYPES_PACKAGE;
------------------------------------------------------------------
```

Abstract Knowledge Types

Blackboard

```
------------------------------------------------------------------------
--
-- Unit      : GENERIC_BLACKBOARD_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 2 January 1992
-- Function  : This package provides the generic operations
--             for the blackboard
--
------------------------------------------------------------------------
generic

  type ITEM_TYPE is private;
  type ITEM_PTR_TYPE is access ITEM_TYPE;
  type LEVEL_INDEX_TYPE is (<>);
  type ITEM_INDEX_TYPE is (<>);

------------------------------------------------------------------------
package GENERIC_BLACKBOARD_PACKAGE is
------------------------------------------------------------------------

    type BLACKBOARD_LEVEL_ARRAY_TYPE is array
         (ITEM_INDEX_TYPE range <>) of ITEM_PTR_TYPE;

    type BLACKBOARD_LEVEL_PTR_TYPE is access
         BLACKBOARD_LEVEL_ARRAY_TYPE;

    type BLACKBOARD_TYPE is array
         (LEVEL_INDEX_TYPE'FIRST .. LEVEL_INDEX_TYPE'LAST) of
         BLACKBOARD_LEVEL_PTR_TYPE;

  ------------------------------------------------------------------------

  procedure CONSTRUCT_BLACKBOARD
  (BLACKBOARD : in out BLACKBOARD_TYPE;
   LEVEL      : in     LEVEL_INDEX_TYPE;
   FROM       : in     ITEM_INDEX_TYPE;
   TO         : in     ITEM_INDEX_TYPE);

  function BLACKBOARD_ITEM
  (BLACKBOARD  : in BLACKBOARD_TYPE;
   LEVEL_INDEX : in LEVEL_INDEX_TYPE;
   ITEM_INDEX  : in ITEM_INDEX_TYPE)
  return ITEM_PTR_TYPE;

  procedure PUT_BLACKBOARD_ITEM
  (BLACKBOARD  : in BLACKBOARD_TYPE;
   ITEM        : in ITEM_PTR_TYPE;
   LEVEL_INDEX : in LEVEL_INDEX_TYPE;
   ITEM_INDEX  : in ITEM_INDEX_TYPE);

------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------
end GENERIC_BLACKBOARD_PACKAGE;
--------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : GENERIC_BLACKBOARD_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 2 January 1992
-- Function  : This package provides the generic operations
--             for the blackboard
--
-------------------------------------------------------------------
package body GENERIC_BLACKBOARD_PACKAGE is
-------------------------------------------------------------------


  -------------------------------------------------------------------
  --
  -- Construct_blackboard builds the blackboard levels to the user
  -- requirement
  --
  -------------------------------------------------------------------
  procedure CONSTRUCT_BLACKBOARD
  -------------------------------------------------------------------
  (BLACKBOARD : in out BLACKBOARD_TYPE;
   LEVEL      : in     LEVEL_INDEX_TYPE;
   FROM       : in     ITEM_INDEX_TYPE;
   TO         : in     ITEM_INDEX_TYPE) is
  -------------------------------------------------------------------
    LEVEL_PTR : BLACKBOARD_LEVEL_PTR_TYPE;
  -------------------------------------------------------------------
  begin

    LEVEL_PTR := new BLACKBOARD_LEVEL_ARRAY_TYPE(FROM .. TO);

    BLACKBOARD(LEVEL) := LEVEL_PTR;
  -------------------------------------------------------------------
  end CONSTRUCT_BLACKBOARD;
  -------------------------------------------------------------------


  -------------------------------------------------------------------
  --
  -- Blackboard_item returns the pointer to an application
  -- blackboard item
  --
  -------------------------------------------------------------------
  function BLACKBOARD_ITEM
  -------------------------------------------------------------------
  (BLACKBOARD  : in BLACKBOARD_TYPE;
   LEVEL_INDEX : in LEVEL_INDEX_TYPE;
   ITEM_INDEX  : in ITEM_INDEX_TYPE)
  return ITEM_PTR_TYPE is
  -------------------------------------------------------------------
  begin

    return BLACKBOARD(LEVEL_INDEX)(ITEM_INDEX);

  -------------------------------------------------------------------
  end BLACKBOARD_ITEM;
  -------------------------------------------------------------------
```

```
   ------------------------------------------------------------------
   -- Put_blackboard item puts the application item to the VDU
   --
   ------------------------------------------------------------------
   procedure PUT_BLACKBOARD_ITEM
   ------------------------------------------------------------------
   (BLACKBOARD   : in BLACKBOARD_TYPE;
    ITEM         : in ITEM_PTR_TYPE;
    LEVEL_INDEX  : in LEVEL_INDEX_TYPE;
    ITEM_INDEX   : in ITEM_INDEX_TYPE) is
   ------------------------------------------------------------------
   begin

     BLACKBOARD(LEVEL_INDEX)(ITEM_INDEX)  := ITEM;

   ------------------------------------------------------------------
   end PUT_BLACKBOARD_ITEM;
   ------------------------------------------------------------------


------------------------------------------------------------------------
end GENERIC_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
```

Logic Knowledge Base

```
------------------------------------------------------------------
--
-- Unit      : LOGIC_KB_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS Shrivenham
-- Date      : 24 July 1991
-- Function : This package provides the operations to build the
--             internal representation of a Prolog knowledge-base.
--
------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     UNCHECKED_DEALLOCATION;
------------------------------------------------------------------
package LOGIC_KB_PACKAGE is
------------------------------------------------------------------

  --  The Knowledge-Base incomplete type definitions to allow use
  --  of KB_NODE_PTR_TYPE in the definition of the instance
  --  structure

  type NODE_TYPE is (HEAD_NODE, FIRST_SUB_GOAL_NODE, SUB_GOAL_NODE,
                     PARAMETER_NODE);
  type KB_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE);
  type KB_NODE_PTR_TYPE is access KB_NODE_RECORD;

------------------------------------------------------------------

  -- The Instance Structures

    subtype VARIABLE_RANGE is NATURAL range 0..40; -- Arbitary!

  -- This is a type definition to limit the number of variables
  -- per clause.  Needed because of the Vax method of using the
  -- maximum value when
  -- eleborating unconstrained arrays with a discriminant.
  -- Otherwise using say POSITIVE will produce a constraint error
  -- in KB_NODE_RECORD since the size of the variables array will be
  -- 1..POSITIVE'LAST - too large for the system!
  -- Note that the latter is acceptable on the Mac SE/30 with
  -- Meridian Ada.

  type VARIABLE_RECORD;
  type INSTANCE_VARIABLES_RECORD(SIZE : VARIABLE_RANGE := 1);
  type INSTANCE_VARIABLES_RECORD_PTR is access
       INSTANCE_VARIABLES_RECORD;

  -- This structure records a variable name together with pointers
  -- to its current instantiated structure in the knowledge-base
  -- and to the record holding the variables associated with that
  -- instantiation.

  type VARIABLE_RECORD is
  record
    TOKEN             : SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.STRING;
    BOUND_STRUCTURE   : KB_NODE_PTR_TYPE;
```

```
  BOUND_VALUE       : INSTANCE_VARIABLES_RECORD_PTR;
end record;

-- These next two structures record all the variables associated
-- with a particular clause.  The array is constrained to the
-- number of variables in the clause

type INSTANCES is array(POSITIVE range <>) of VARIABLE_RECORD;

type INSTANCE_VARIABLES_RECORD(SIZE : VARIABLE_RANGE := 1) is
record
  VARIABLES         : INSTANCES(1 .. SIZE);
  VARIABLE_COUNT    : NATURAL := 0;
  LEVEL             : NATURAL := 0;
end record;

-------------------------------------------------------------------

-- Complete knowledge-base record definitions

type KB_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE) is
record
  NAME              : SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.STRING;
  ARITY             : NATURAL := 0;
  INDEX             : VARIABLE_RANGE := 0;
  NEXT_PARAMETER    : KB_NODE_PTR_TYPE;
  NEXT_GOAL         : KB_NODE_PTR_TYPE;
    case KIND is
      when HEAD_NODE            =>
        CLAUSE              : SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
        INSTANCE_TEMPLATE : INSTANCE_VARIABLES_RECORD;
        NEXT_CLAUSE         : KB_NODE_PTR_TYPE;
      when FIRST_SUB_GOAL_NODE =>
        OR_SUB_GOAL         : KB_NODE_PTR_TYPE;
      when SUB_GOAL_NODE | PARAMETER_NODE       => null;
    end case;
end record;

type KB_RECORD is
record
  FIRST : KB_NODE_PTR_TYPE;
  LAST  : KB_NODE_PTR_TYPE;
end record;

-------------------------------------------------------------------

 procedure DISPOSE is new UNCHECKED_DEALLOCATION
                          (KB_NODE_RECORD,
                           KB_NODE_PTR_TYPE);
-------------------------------------------------------------------

procedure BUILD(KB         : in out KB_RECORD;
                FILE_NAME : in     STANDARD.STRING);

  procedure ASSERT
  (IN_CLAUSE  : in     STANDARD.
                       STRING;
```

```
     KB         : in out KB_RECORD;
     AT_BACK_OF : in     BOOLEAN := TRUE);

    procedure RETRACT
    (CLAUSE : in     STANDARD.
                     STRING;
     KB     : in out KB_RECORD);

  procedure PRINT_CLAUSE(CLAUSE : in KB_NODE_PTR_TYPE);
  procedure PRINT_PREDICATE(PRED : in KB_NODE_PTR_TYPE);
  procedure PRINT(KB : in KB_RECORD);
  procedure TEST;

-------------------------------------------------------------------------
end LOGIC_KB_PACKAGE;
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Title     : LOGIC_KB_PACKAGE package body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS Shrivenham
-- Date      : 24 July 1991
-- Function  : This package provides the operations to build the
--             internal representation of a Prolog knowledge-base
--
------------------------------------------------------------------
with GENERIC_LIST_PACKAGE;
------------------------------------------------------------------
package body LOGIC_KB_PACKAGE is
------------------------------------------------------------------

  package INTEGER_TEXT_IO is new TEXT_IO.INTEGER_IO(INTEGER);
  package VARIABLES_QUEUE is new
                             GENERIC_LIST_PACKAGE
                             (SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.STRING,
                              SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.IS_EQUAL);

------------------------------------------------------------------

  -- An object of Input_clause_record is used during input from the
  -- external Prolog source file

  type INPUT_CLAUSE_RECORD is
    record

     SPACED_VALUE : SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    STRING;
     SHORT_VALUE  : SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    STRING;

     INDEX : NATURAL := 1;

    end record;

  ------------------------------------------------------------------
  --
  -- Print_structure is a diagnostic print subprogram which
  -- recursively displays the  bracketed structures in a clause
  --
  ------------------------------------------------------------------
  procedure PRINT_STRUCTURE
  ------------------------------------------------------------------
  (NODE_PTR : in KB_NODE_PTR_TYPE) is
  ------------------------------------------------------------------
      TEMP : KB_NODE_PTR_TYPE := NODE_PTR;
  ------------------------------------------------------------------
  begin
    TEXT_IO.PUT("(");

    while TEMP /= null
    loop
```

```
      INTEGER_TEXT_IO.PUT(TEMP.ARITY,2);
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.PUT(TEMP.NAME);
      INTEGER_TEXT_IO.PUT(TEMP.INDEX, 1);

      if TEMP.NEXT_PARAMETER /= null then

        PRINT_STRUCTURE(TEMP.NEXT_PARAMETER);

      end if;

      TEMP := TEMP.NEXT_GOAL;

      if TEMP /= null then

        TEXT_IO.PUT(",");

      end if;

    end loop;

    TEXT_IO.PUT(")");
  ------------------------------------------------------------------
  exception
    when OTHERS =>
      TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                       "PRINT_STRUCTURE");
  ------------------------------------------------------------------
  end PRINT_STRUCTURE;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  --
  -- Print_clause displays a single clause using print_structure to
  -- deal with brackets.   For each token in the clause the token
  -- arity is displayed first followed by the token name and
  -- terminated with the index of the token in the
  -- variable template.   An atom will have an index of 0 indicating
  -- that it is not entered in the variable template
  --
  ------------------------------------------------------------------
  procedure PRINT_CLAUSE
  ------------------------------------------------------------------
  (CLAUSE : in KB_NODE_PTR_TYPE) is
  ------------------------------------------------------------------
    TEMP     : KB_NODE_PTR_TYPE;
    ROOT_PTR : KB_NODE_PTR_TYPE;
  ------------------------------------------------------------------
  begin

    -- print head

    INTEGER_TEXT_IO.PUT(CLAUSE.ARITY,2);
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.PUT(CLAUSE.NAME);
    INTEGER_TEXT_IO.PUT(CLAUSE.INDEX, 1);

    if CLAUSE.NEXT_PARAMETER /= null then
```

```
    -- print head parameters

    PRINT_STRUCTURE(CLAUSE.NEXT_PARAMETER);

  end if;

  TEMP := CLAUSE.NEXT_GOAL;

  if TEMP /= null then

    TEXT_IO.PUT(":-");

    loop

      -- print rest of rule

      ROOT_PTR := TEMP;

      -- Root_ptr indicates the point at which 'or' predicates
      -- would be attached to the knowledge-base

      while TEMP /= null
      loop

        -- print head of predicate

        INTEGER_TEXT_IO.PUT(TEMP.ARITY,2);
        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.PUT(TEMP.NAME);
        INTEGER_TEXT_IO.PUT(TEMP.INDEX, 1);

        if TEMP.NEXT_PARAMETER /= null then

          PRINT_STRUCTURE(TEMP.NEXT_PARAMETER);

        end if;

        TEMP := TEMP.NEXT_GOAL;

        if TEMP /= null then

          TEXT_IO.PUT(",");

        end if;

      end loop;

      -- check to see if there are any 'or' predicates in the
      -- knowledge_base

      exit when ROOT_PTR.OR_SUB_GOAL = null;

      TEXT_IO.PUT(";");
      TEMP := ROOT_PTR.OR_SUB_GOAL;

    end loop;

  end if;
-------------------------------------------------------------------
exception
```

```
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "PRINT_PREDICATE");
-------------------------------------------------------------------
end PRINT_CLAUSE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Print_predicate displays a single predicate using
-- print_structure to deal with brackets
--
-------------------------------------------------------------------
procedure PRINT_PREDICATE
-------------------------------------------------------------------
(PRED : in KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------
begin


   INTEGER_TEXT_IO.PUT(PRED.ARITY,2);
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.PUT(PRED.NAME);
   INTEGER_TEXT_IO.PUT(PRED.INDEX, 1);

   if PRED.NEXT_PARAMETER /= null then

      PRINT_STRUCTURE(PRED.NEXT_PARAMETER);

   end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "PRINT_PREDICATE");
-------------------------------------------------------------------
end PRINT_PREDICATE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Print the knowledge base
--
-------------------------------------------------------------------
procedure PRINT
(KB : in KB_RECORD) is
-------------------------------------------------------------------
  CLAUSE_PTR : KB_NODE_PTR_TYPE := KB.FIRST;
-------------------------------------------------------------------
begin

  while CLAUSE_PTR /= null
  loop

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    PUT
    (THE_STRING => CLAUSE_PTR.
```

```
                CLAUSE);

    TEXT_IO.NEW_LINE;

    CLAUSE_PTR := CLAUSE_PTR.
                  NEXT_CLAUSE;

  end loop;
-------------------------------------------------------------------
end PRINT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Get uses the operations provided in dynamic
-- string to extract a single clause from the knowledge-base
-- source file
--
-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(CLAUSE   : in out INPUT_CLAUSE_RECORD;
 FILE     : in out TEXT_IO.FILE_TYPE) is
-------------------------------------------------------------------
  CHAR : CHARACTER;
  -----------------------------------------------------------------
  --
  -- The operation Clear_this clears the content of a
  -- dynamic_string
  --
  -----------------------------------------------------------------
  procedure CLEAR_THIS
  -----------------------------------------------------------------
  (CLAUSE : in out INPUT_CLAUSE_RECORD) is
  -----------------------------------------------------------------
  begin

    if not SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.IS_NULL(CLAUSE.SPACED_VALUE) then
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.CLEAR(CLAUSE.SPACED_VALUE);
    end if;

    if not SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.IS_NULL(CLAUSE.SHORT_VALUE) then
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.CLEAR(CLAUSE.SHORT_VALUE);
    end if;

    CLAUSE.INDEX := 1;
  -----------------------------------------------------------------
  end CLEAR_THIS;
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  --
  -- The operation ADD appends a character to the current clause
  --
```

```
   ------------------------------------------------------------------
   procedure ADD
   ------------------------------------------------------------------
   (CHAR   : in     CHARACTER;
    CLAUSE : in out INPUT_CLAUSE_RECORD) is
   ------------------------------------------------------------------
   begin

     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     APPEND
     (THE_ITEM      => CHAR,
      TO_THE_STRING => CLAUSE.SPACED_VALUE);
   ------------------------------------------------------------------
   end ADD;
   ------------------------------------------------------------------

------------------------------------------------------------------
begin

  CLEAR_THIS(CLAUSE);
  TEXT_IO.GET(FILE, CHAR);

  while CHAR /= '.'
  loop

    -- Add each character and delimit each word/operation with
    -- spaces

    case CHAR is
      when 'a' .. 'z' |
           'A' .. 'Z' |
           '0' .. '9' |
           ' '        |
           '_'              => ADD(CHAR, CLAUSE);
                               if CHAR /= ' ' then
                                 SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING.
                                 APPEND
                                 (THE_ITEM      => CHAR,
                                  TO_THE_STRING => CLAUSE.
                                                   SHORT_VALUE);
                               end if;

      when '(' | ')' |
           ',' | ';' | '!'  => ADD(' ', CLAUSE);
                               ADD(CHAR, CLAUSE);
                               ADD(' ', CLAUSE);

                               SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               APPEND
                               (THE_ITEM      => CHAR,
                                TO_THE_STRING => CLAUSE.
                                                 SHORT_VALUE);

      when ':'             => ADD(' ', CLAUSE);
                               ADD(CHAR, CLAUSE);

                               SYSTEM_TYPES_PACKAGE.
```

```
                                DYNAMIC_STRING.
                                APPEND
                                (THE_ITEM       => CHAR,
                                 TO_THE_STRING => CLAUSE.
                                 SHORT_VALUE);

                                TEXT_IO.GET(FILE, CHAR);
                                ADD(CHAR, CLAUSE);
                                ADD(' ', CLAUSE);

                                SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                APPEND
                                (THE_ITEM       => CHAR,
                                 TO_THE_STRING => CLAUSE.
                                                  SHORT_VALUE);

      when '=' | '\' | '<'   => ADD(' ', CLAUSE);
                                ADD(CHAR, CLAUSE);

                                SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                APPEND
                                (THE_ITEM       => CHAR,
                                 TO_THE_STRING => CLAUSE.
                                                  SHORT_VALUE);

                                TEXT_IO.GET(FILE, CHAR);
                                while CHAR /= ' '
                                loop
                                  ADD(CHAR, CLAUSE);

                                  SYSTEM_TYPES_PACKAGE.
                                  DYNAMIC_STRING.
                                  APPEND
                                  (THE_ITEM       => CHAR,
                                   TO_THE_STRING => CLAUSE.
                                                    SHORT_VALUE);

                                  TEXT_IO.GET(FILE, CHAR);
                                end loop;
                                ADD(' ', CLAUSE);

      when others             => null;
    end case;

    TEXT_IO.GET(FILE, CHAR);

  end loop;

  -- Add the period

  ADD(' ', CLAUSE);
  ADD(CHAR, CLAUSE);
  ADD(' ', CLAUSE);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  APPEND
  (THE_ITEM       => CHAR,
```

```
   TO_THE_STRING => CLAUSE.SHORT_VALUE);
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "GET_1");
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Get extracts a single token from the current
-- clause
--
-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(TOKEN  : in out SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 CLAUSE : in out INPUT_CLAUSE_RECORD) is
-------------------------------------------------------------------
begin

  if not SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.IS_NULL(TOKEN) then
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.CLEAR(TOKEN);
  end if;

  -- Clear spaces

  while SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.ITEM_OF
        (CLAUSE.SPACED_VALUE, CLAUSE.INDEX) = ' '
  loop
    CLAUSE.INDEX := CLAUSE.INDEX + 1;
  end loop;

  -- Form token

  while SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        ITEM_OF(CLAUSE.SPACED_VALUE,CLAUSE.INDEX) /= ' '
  loop
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.APPEND( SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.ITEM_OF
                           (CLAUSE.SPACED_VALUE,
                            CLAUSE.INDEX),
                            TOKEN);
    CLAUSE.INDEX := CLAUSE.INDEX + 1;
  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "GET_2");
```

```
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Form constructs the head of the current clause
--
-------------------------------------------------------------------
procedure FORM
-------------------------------------------------------------------
(HEAD  : in out KB_NODE_PTR_TYPE;
 TOKEN : in     SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.STRING) is
-------------------------------------------------------------------
begin

 HEAD := new KB_NODE_RECORD(KIND => HEAD_NODE);
 SYSTEM_TYPES_PACKAGE.
 DYNAMIC_STRING.COPY(TOKEN, HEAD.NAME);
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "FORM");
-------------------------------------------------------------------
end FORM;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Add_to_front attaches the head node to the
-- knowledge-base structure
--
-------------------------------------------------------------------
procedure ADD_TO_FRONT
-------------------------------------------------------------------
(HEAD : in     KB_NODE_PTR_TYPE;
 KB   : in out KB_RECORD) is
-------------------------------------------------------------------
begin

  if (KB.FIRST = KB.LAST) and (KB.FIRST = null) then

    -- first clause

    KB.FIRST := HEAD;
    KB.LAST := HEAD;

  else

    -- subsequent clauses

    HEAD.NEXT_CLAUSE := KB.FIRST;
    KB.FIRST := HEAD;

  end if;
-------------------------------------------------------------------
```

```
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "ADD_TO_FRONT");
-------------------------------------------------------------------
end ADD_TO_FRONT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Add_to_back attaches the head node to the
-- knowledge-base structure
--
-------------------------------------------------------------------
procedure ADD_TO_BACK
-------------------------------------------------------------------
(HEAD : in     KB_NODE_PTR_TYPE;
 KB   : in out KB_RECORD) is
-------------------------------------------------------------------
begin

  if (KB.FIRST = KB.LAST) and (KB.FIRST = null) then

    -- first clause

    KB.FIRST := HEAD;
    KB.LAST := HEAD;

  else

    -- subsequent clauses

    KB.LAST.NEXT_CLAUSE := HEAD;
    KB.LAST := HEAD;

  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "ADD_TO_BACK");
-------------------------------------------------------------------
end ADD_TO_BACK;
-------------------------------------------------------------------


-------------------------------------------------------------------

--
-- The operation Add_first_goal connects the first node to the
-- head node in the current clause
--
-------------------------------------------------------------------
procedure ADD_FIRST_GOAL
-------------------------------------------------------------------
(TOKEN     : in     SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.STRING;
 PREV_HEAD : in out KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------
```

```
   -- create new node

   GOAL : KB_NODE_PTR_TYPE := new KB_NODE_RECORD
                                    (KIND => FIRST_SUB_GOAL_NODE);
-------------------------------------------------------------------
begin

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.COPY(TOKEN, GOAL.NAME);
   PREV_HEAD.NEXT_GOAL := GOAL;
   PREV_HEAD := GOAL;
-------------------------------------------------------------------
exception
   when OTHERS =>
     TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                      "ADD_FIRST_GOAL");
-------------------------------------------------------------------
end ADD_FIRST_GOAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Add_goal connects the current node to the
-- previous node in the current clause
--
-------------------------------------------------------------------
procedure ADD_GOAL
-------------------------------------------------------------------
(TOKEN     : in     SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.STRING;
 PREV_HEAD : in out KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------

   -- create new node

   GOAL : KB_NODE_PTR_TYPE := new KB_NODE_RECORD
                                    (KIND => SUB_GOAL_NODE);
-------------------------------------------------------------------
begin

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.COPY(TOKEN, GOAL.NAME);
   PREV_HEAD.NEXT_GOAL := GOAL;
   PREV_HEAD := GOAL;
-------------------------------------------------------------------
exception
   when OTHERS =>
     TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                      "ADD_GOAL");
-------------------------------------------------------------------
end ADD_GOAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Add_first_parameter connects the first parameter
-- of the current predicate to the current clause
```

```
--
-------------------------------------------------------------------
procedure ADD_FIRST_PARAMETER
-------------------------------------------------------------------
(TOKEN      : in     SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.STRING;
 PREV_HEAD : in out KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------

  PARAMETER : KB_NODE_PTR_TYPE := new KB_NODE_RECORD
                                      (KIND => PARAMETER_NODE);
-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.COPY(TOKEN, PARAMETER.NAME);
  PREV_HEAD.NEXT_PARAMETER := PARAMETER;
  PREV_HEAD := PARAMETER;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "ADD_FIRST_PARAMETER");
-------------------------------------------------------------------
end ADD_FIRST_PARAMETER;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Add_next_parameter connects subsequent parameters
-- of a predicate to the current clause
--
-------------------------------------------------------------------
procedure ADD_NEXT_PARAMETER
-------------------------------------------------------------------
 (TOKEN      : in     SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.STRING;
  PREV_HEAD : in out KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  PARAMETER : KB_NODE_PTR_TYPE := new KB_NODE_RECORD
                                     (KIND => PARAMETER_NODE);
-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.COPY(TOKEN, PARAMETER.NAME);
  PREV_HEAD.NEXT_GOAL := PARAMETER;
  PREV_HEAD := PARAMETER;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "ADD_NEXT_PARAMETER");
-------------------------------------------------------------------
end ADD_NEXT_PARAMETER;
-------------------------------------------------------------------


-------------------------------------------------------------------
```

```
--
-- The Add_or operation connects an 'or' subgoal to the clause
-- field at the start of any previous subgoals.   This point is
-- indicated by root
--
-------------------------------------------------------------------
procedure ADD_OR
-------------------------------------------------------------------
(TOKEN : in     SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.STRING;
 ROOT  : in out KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------

 GOAL : KB_NODE_PTR_TYPE := new
                            KB_NODE_RECORD(FIRST_SUB_GOAL_NODE);
-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.COPY(TOKEN, GOAL.NAME);
  ROOT.OR_SUB_GOAL := GOAL;
  ROOT := GOAL;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "ADD_OR");
-------------------------------------------------------------------
end ADD_OR;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- The operation Check_if_variable determines whether the current
-- node element is a variable.  If it is then it is added to the
-- list of variables for the current clause
--
-------------------------------------------------------------------
procedure CHECK_IF_VARIABLE
-------------------------------------------------------------------
(NODE           : in     KB_NODE_PTR_TYPE;
 VARIABLES      : in out VARIABLES_QUEUE.LIST_TYPE;
 VARIABLE_COUNT : in out VARIABLE_RANGE) is
-------------------------------------------------------------------
  INDEX : VARIABLE_RANGE := 0;
-------------------------------------------------------------------
begin

  if  ( SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.ITEM_OF(NODE.NAME, 1) in 'A' .. 'Z') or
      (SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.ITEM_OF(NODE.NAME, 1) = '_') then

    -- it is a variable.
    -- Check whether it is already recorded for this clause

    INDEX := VARIABLES_QUEUE.POSITION_IN(VARIABLES, NODE.NAME);

    if (INDEX = 0) or (SYSTEM_TYPES_PACKAGE.
```

```
                        DYNAMIC_STRING.ITEM_OF
                        (NODE.NAME, 1) = '_') then

      -- A zero index signifies not recorded.
      -- Record all '_' since these are treated as separate
      -- variables within
      -- a clause

      VARIABLES_QUEUE.PUT_ON_BACK_OF(VARIABLES, NODE.NAME);
      VARIABLE_COUNT := VARIABLE_COUNT + 1;
      NODE.INDEX := VARIABLE_COUNT;

    else

      -- Already recorded so ignore it

      NODE.INDEX := INDEX;

    end if;

  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "CHECK_IF_VARIABLE");
-------------------------------------------------------------------
end CHECK_IF_VARIABLE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Form_structure recursively connects bracketed elements to the
-- current clause.
-- It deals with parameters and structures
-- Prev_head.      A pointer to the previous node to which this
--                 parameter will be connected
-- First_token.    The first element of the structure being formed
-- Clause.         The current clause being constructed
-- Arity_1.        The arity of current bracketed level
-- Variables.      A record of the variables found so far in
--                 current clause
-- Variable_count. The number of variables found so far in current
--                 clause
-- Temp_ptr.       Tracks progress along current level
-- Token.          Current atom/variable
-- Op.             Operator prior to current Token
-- Arity_2         Arity of next level of brackets if present
--
-------------------------------------------------------------------
procedure FORM_STRUCTURE
-------------------------------------------------------------------
(PREV_HEAD      : in out KB_NODE_PTR_TYPE;
 FIRST_TOKEN    : in     SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.STRING;
 CLAUSE         : in out INPUT_CLAUSE_RECORD;
 ARITY_1        : in out NATURAL;
 VARIABLES      : in out VARIABLES_QUEUE.LIST_TYPE;
 VARIABLE_COUNT : in out VARIABLE_RANGE) is
```

```
------------------------------------------------------------------
  TEMP_PTR : KB_NODE_PTR_TYPE := PREV_HEAD;
  TOKEN    : SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.STRING;
  OP       : SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.STRING;
  ARITY_2  : NATURAL := 0;
------------------------------------------------------------------
begin

  -- Connect first element

  ADD_FIRST_PARAMETER(FIRST_TOKEN, TEMP_PTR);
  ARITY_1 := ARITY_1 + 1;

  CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

  GET(OP, CLAUSE);
  while not SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.IS_EQUAL(OP, ")")
  loop
    GET(TOKEN, CLAUSE);

    if SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.IS_EQUAL(OP, ",") then

      ADD_NEXT_PARAMETER(TOKEN, TEMP_PTR);

      -- Increase arity of this structure

      ARITY_1 := ARITY_1 + 1;

      CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

    elsif SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.IS_EQUAL(OP, "(") then

      -- Another opening bracket therefore recurse to next level

      FORM_STRUCTURE( TEMP_PTR,
                      TOKEN,
                      CLAUSE,
                      ARITY_2,
                      VARIABLES,
                      VARIABLE_COUNT);

      -- Deposit arity of the structure just completed

      TEMP_PTR.ARITY := ARITY_2;

      -- Could have added this detail to the token e.g
      -- DYNAMIC_STRING.PREPEND
      -- (NATURAL'IMAGE(ARITY_2), TEMP_PTR.NAME);

      ARITY_2 := 0;

    end if;

  GET(OP, CLAUSE);
```

```
  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "FORM_STRUCTURE");
-------------------------------------------------------------------
end FORM_STRUCTURE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Prefix_operation forces the infix built-in operations into
-- prefix form
--
-------------------------------------------------------------------
procedure PREFIX_OPERATION
(OP_NODE         : in     KB_NODE_PTR_TYPE;
 OP              : in     SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.STRING;
 LEFT_HAND_SIDE :     out KB_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  LEFT_HAND_SIDE_PARAMETERS : KB_NODE_PTR_TYPE :=
                              OP_NODE.NEXT_PARAMETER;
  LEFT_HAND_SIDE_HEAD       : KB_NODE_PTR_TYPE :=
                              new KB_NODE_RECORD
                                  (KIND => SUB_GOAL_NODE);
-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.COPY(OP_NODE.NAME, LEFT_HAND_SIDE_HEAD.NAME);
  LEFT_HAND_SIDE_HEAD.ARITY := OP_NODE.ARITY;
  LEFT_HAND_SIDE_HEAD.INDEX := OP_NODE.INDEX;
  LEFT_HAND_SIDE_HEAD.NEXT_PARAMETER := LEFT_HAND_SIDE_PARAMETERS;
  OP_NODE.NEXT_PARAMETER := LEFT_HAND_SIDE_HEAD;
  OP_NODE.ARITY := 2;
  OP_NODE.INDEX := 0;
  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.COPY(OP, OP_NODE.NAME);
  LEFT_HAND_SIDE := LEFT_HAND_SIDE_HEAD;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in LOGIC_KB_PACKAGE at " &
                     "PREFIX_OPERATION");
-------------------------------------------------------------------
end PREFIX_OPERATION;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Assert
--
-------------------------------------------------------------------
procedure ASSERT
(IN_CLAUSE  : in     STANDARD.
                     STRING;
```

```
KB          : in out KB_RECORD;
AT_BACK_OF  : in     BOOLEAN := TRUE) is
-------------------------------------------------------------------
 CLAUSE          : INPUT_CLAUSE_RECORD;
 TOKEN           : SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.STRING;
 OP              : SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.STRING;
 NEW_STRING      : SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.STRING;
 VARIABLES       : VARIABLES_QUEUE.LIST_TYPE;
 ROOT_PTR        : KB_NODE_PTR_TYPE;
 TEMP_PTR        : KB_NODE_PTR_TYPE;
 CLAUSE_HEAD     : KB_NODE_PTR_TYPE;
 ARITY           : NATURAL := 0;
 VARIABLE_COUNT  : VARIABLE_RANGE := 0;

 SYNTAX_ERROR    : exception;


 -------------------------------------------------------------------
 procedure GET
 -------------------------------------------------------------------
 (CLAUSE       : in out INPUT_CLAUSE_RECORD;
  IN_CLAUSE    : in     STANDARD.STRING) is

   CHAR : CHARACTER;
   COUNT : NATURAL := 1;

   -------------------------------------------------------------------
   --
   -- The operation Clear_this clears the content of a
   -- dynamic_string
   --
   -------------------------------------------------------------------
   procedure CLEAR_THIS
   -------------------------------------------------------------------
   (CLAUSE : in out INPUT_CLAUSE_RECORD) is
   -------------------------------------------------------------------
   begin

    if not SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.IS_NULL(CLAUSE.SPACED_VALUE) then
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.CLEAR(CLAUSE.SPACED_VALUE);
    end if;

    CLAUSE.INDEX := 1;
   -------------------------------------------------------------------
   end CLEAR_THIS;
   -------------------------------------------------------------------


   -------------------------------------------------------------------
   --
   -- The operation ADD appends a character to the current clause
   --
   -------------------------------------------------------------------
   procedure ADD
   -------------------------------------------------------------------
```

```
    (CHAR   : in     CHARACTER;
     CLAUSE : in out INPUT_CLAUSE_RECORD) is
    ------------------------------------------------------------
    begin

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.APPEND(CHAR, CLAUSE.SPACED_VALUE);
    ------------------------------------------------------------
    end ADD;
    ------------------------------------------------------------

  ------------------------------------------------------------
  begin

    CLEAR_THIS(CLAUSE);

    while COUNT <= IN_CLAUSE'LENGTH
    loop

      -- Add each character and delimit each word/operation
      -- with spaces

      CHAR := IN_CLAUSE(COUNT);

      case CHAR is
        when 'a' .. 'z' |
             'A' .. 'Z' |
             '0' .. '9' |
             ' '        |
             '_'              => ADD(CHAR, CLAUSE);
                                 if CHAR /= ' ' then
                                   SYSTEM_TYPES_PACKAGE.
                                   DYNAMIC_STRING.
                                   APPEND
                                   (THE_ITEM      => CHAR,
                                    TO_THE_STRING => CLAUSE.
                                    SHORT_VALUE);
                                 end if;

        when '(' | ')' |
             ',' | ';' | '!'  => ADD(' ', CLAUSE);
                                 ADD(CHAR, CLAUSE);
                                 ADD(' ', CLAUSE);

                                 SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING.
                                 APPEND
                                 (THE_ITEM      => CHAR,
                                  TO_THE_STRING => CLAUSE.
                                                   SHORT_VALUE);

        when ':'              => ADD(' ', CLAUSE);
                                 ADD(CHAR, CLAUSE);

                                 SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING.
                                 APPEND
                                 (THE_ITEM      => CHAR,
                                  TO_THE_STRING => CLAUSE.
                                                   SHORT_VALUE);
```

```
                                    COUNT := COUNT + 1;
                                    CHAR := IN_CLAUSE(COUNT);
                                    ADD(CHAR, CLAUSE);
                                    ADD(' ', CLAUSE);

                                    SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    APPEND
                                    (THE_ITEM       => CHAR,
                                     TO_THE_STRING => CLAUSE.
                                                      SHORT_VALUE);

      when '=' | '\' | '<'      => ADD(' ', CLAUSE);
                                    ADD(CHAR, CLAUSE);
                                    COUNT := COUNT + 1;
                                    CHAR := IN_CLAUSE(COUNT);

                                    SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    APPEND
                                    (THE_ITEM       => CHAR,
                                     TO_THE_STRING => CLAUSE.
                                                      SHORT_VALUE);

                                    while CHAR /= ' '
                                    loop
                                      ADD(CHAR, CLAUSE);

                                      SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING.
                                      APPEND
                                      (THE_ITEM       => CHAR,
                                       TO_THE_STRING =>
                                       CLAUSE.
                                       SHORT_VALUE);

                                      COUNT := COUNT + 1;
                                      CHAR := IN_CLAUSE(COUNT);
                                    end loop;
                                    ADD(' ', CLAUSE);

      when '.'                  => ADD(' ', CLAUSE);
                                    ADD(CHAR, CLAUSE);
                                    ADD(' ', CLAUSE);

                                    SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    APPEND
                                    (THE_ITEM       => CHAR,
                                     TO_THE_STRING => CLAUSE.
                                                      SHORT_VALUE);

      when others               => null;

    end case;

  COUNT := COUNT + 1;

  end loop;
```

```
   ------------------------------------------------------------------
   end GET;
   ------------------------------------------------------------------

 ------------------------------------------------------------------
begin

    -- Form head of current predicate

    GET(CLAUSE, IN_CLAUSE);

    GET(TOKEN, CLAUSE);

    ARITY := 0;

    FORM(CLAUSE_HEAD, TOKEN);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_STRING => CLAUSE.SHORT_VALUE,
     TO_THE_STRING   => CLAUSE_HEAD.CLAUSE);

    TEMP_PTR := CLAUSE_HEAD;

    if AT_BACK_OF then
      ADD_TO_BACK(CLAUSE_HEAD, KB);
    else
      ADD_TO_FRONT(CLAUSE_HEAD, KB);
    end if;

    GET(OP, CLAUSE);

    while not SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.IS_EQUAL(OP, ".")
    loop

      GET(TOKEN, CLAUSE);

      if SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.IS_EQUAL(OP, ":-") then

        -- A rule therefore reset pointer to head of clause so
        -- that subgoals can be added

        TEMP_PTR := CLAUSE_HEAD;
        ADD_FIRST_GOAL(TOKEN, TEMP_PTR);

        ROOT_PTR := TEMP_PTR;
        ARITY := 0;

        CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

      elsif SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.IS_EQUAL(OP, ",") then

        ADD_GOAL(TOKEN, TEMP_PTR);

        CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);
```

```
elsif SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.IS_EQUAL(OP, ";") then

  ADD_OR(TOKEN, ROOT_PTR);
  TEMP_PTR := ROOT_PTR;

  CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

elsif SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.IS_EQUAL(OP, "(") then

  FORM_STRUCTURE( TEMP_PTR,
                  TOKEN,
                  CLAUSE,
                  ARITY,
                  VARIABLES,
                  VARIABLE_COUNT);

  -- Set arity of this predicate
  -- then reset ready for next time around if required

  TEMP_PTR.ARITY := ARITY;
  ARITY := 0;

  -- Could have added this detail to the token e.g
  -- DYNAMIC_STRING.PREPEND
  -- (NATURAL'IMAGE(ARITY), TEMP_PTR.NAME);

else

  raise SYNTAX_ERROR;

end if;

GET(OP, CLAUSE);

-- Is this an infix operation?

if SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.IS_EQUAL(OP, "\=") or else
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.IS_EQUAL(OP, "is") or else
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.IS_EQUAL(OP, "<") then

  declare

    PARAMETER_PTR : KB_NODE_PTR_TYPE;

  begin

    PREFIX_OPERATION(TEMP_PTR,
                     OP,
                     PARAMETER_PTR);

    GET(TOKEN, CLAUSE);

    ADD_GOAL(TOKEN, PARAMETER_PTR);
    CHECK_IF_VARIABLE
    (PARAMETER_PTR,
```

```
            VARIABLES,
            VARIABLE_COUNT);
          GET(OP, CLAUSE);

          if SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.IS_EQUAL(OP, "(") then
            GET(TOKEN, CLAUSE);

            FORM_STRUCTURE(PARAMETER_PTR,
                           TOKEN,
                           CLAUSE,
                           ARITY,
                           VARIABLES,
                           VARIABLE_COUNT);
            PARAMETER_PTR.ARITY := ARITY;
            ARITY := 0;
            GET(OP, CLAUSE);

          end if;
        end;
      end if;

    end loop;

    -- Transfer variables to clause variable template in clause
    -- head

    if VARIABLE_COUNT /= 0 then

      -- Set discriminant to correct size

      CLAUSE_HEAD.INSTANCE_TEMPLATE :=
      (SIZE           => VARIABLE_COUNT,
       VARIABLES      => (1 .. VARIABLE_COUNT
                          => (NEW_STRING,
                              null,
                              null)),
       VARIABLE_COUNT => VARIABLE_COUNT,
       LEVEL          => 0);

      -- transfer the variable details

      for COUNT in 1 .. VARIABLE_COUNT
      loop
        CLAUSE_HEAD.INSTANCE_TEMPLATE.VARIABLES(COUNT).TOKEN :=
        VARIABLES_QUEUE.FRONT_OF(VARIABLES);
        VARIABLES_QUEUE.DELETE_FRONT_OF(VARIABLES);
      end loop;

      CLAUSE_HEAD.INSTANCE_TEMPLATE.VARIABLE_COUNT :=
      VARIABLE_COUNT;
    end if;
-------------------------------------------------------------------
exception
  when SYNTAX_ERROR =>
    TEXT_IO.PUT_LINE("SYNTAX_ERROR in LOGIC_KS_PACKAGE at" &
                     "ASSERT");
-------------------------------------------------------------------
end ASSERT;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure RETRACT
-------------------------------------------------------------------
(CLAUSE : in     STANDARD.
                 STRING;
 KB     : in out KB_RECORD) is
-------------------------------------------------------------------

  CLAUSE_PTR    : KB_NODE_PTR_TYPE;
  PREVIOUS_PTR : KB_NODE_PTR_TYPE;

  CLAUSE_NOT_FOUND : exception;

  -----------------------------------------------------------------
  procedure DISPOSE_OF
  -----------------------------------------------------------------
  (CLAUSE_PTR : in out KB_NODE_PTR_TYPE) is
  -----------------------------------------------------------------
  begin
    if CLAUSE_PTR = null then
      return;
    else
      DISPOSE_OF(CLAUSE_PTR.NEXT_GOAL);
      if CLAUSE_PTR.NEXT_PARAMETER /= null then
        DISPOSE_OF(CLAUSE_PTR.NEXT_PARAMETER);
      end if;
      DISPOSE(CLAUSE_PTR);
    end if;
  -----------------------------------------------------------------
  end DISPOSE_OF;
  -----------------------------------------------------------------


-------------------------------------------------------------------
begin

  if KB.FIRST = null then
    raise CLAUSE_NOT_FOUND;
  else
    CLAUSE_PTR := KB.FIRST;
    loop
      if SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL
         (LEFT  => CLAUSE,
          RIGHT => CLAUSE_PTR.CLAUSE) then
        exit;
      else
        PREVIOUS_PTR := CLAUSE_PTR;
        CLAUSE_PTR := CLAUSE_PTR.NEXT_CLAUSE;
        exit when CLAUSE_PTR = null;
      end if;
    end loop;
  end if;

  if CLAUSE_PTR = null then
    raise CLAUSE_NOT_FOUND;
  elsif KB.FIRST = KB.LAST then -- Only one clause
    DISPOSE_OF(KB.FIRST);
    KB.LAST := null;
```

```
  elsif KB.FIRST = CLAUSE_PTR then -- First clause
    KB.FIRST := CLAUSE_PTR.NEXT_CLAUSE;
    DISPOSE_OF(CLAUSE_PTR);
  elsif KB.LAST = CLAUSE_PTR then -- Last clause
    KB.LAST := PREVIOUS_PTR;
    DISPOSE_OF(CLAUSE_PTR);
  else -- A middle clause
    PREVIOUS_PTR.NEXT_CLAUSE := CLAUSE_PTR.NEXT_CLAUSE;
    DISPOSE_OF(CLAUSE_PTR);
  end if;
-------------------------------------------------------------------
exception
  when CLAUSE_NOT_FOUND =>
    TEXT_IO.PUT_LINE("Exception CLAUSE_NOT_FOUND raised in" &
                     "LOGIC_KB_PACKAGE" &
                     " at RETRACT");
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS raised in LOGIC_KB_PACKAGE"
                     &
                     " at RETRACT");
-------------------------------------------------------------------
end RETRACT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Build constructs the knowledge-base one clause at a time
--
-- Kb.              The completed knowledge-base
-- File_name.       Name of external source file containg Prolog
--                  source
--
-- Kb_file.         Internal file name
-- Clause.          Current clause
-- Token.           Current atom/variable
-- Op.              Current operator
-- New_string.      A dummy dynamic string used in forming the
--                  variables template
-- Variables.       A list of the variables in the current clause
-- Root_ptr.        Indicates the position at which 'or' subgoals
--                  would be
--                  attached to the current clause
-- Temp_ptr.        Tracks the current position within the current
--                  clause
-- Clause_head.     Indicates the head of the current clause
-- Arity.           The arity of current predicate in the current
--                  clause
-- Variable_count.  The number of variables in the current clause
-------------------------------------------------------------------
procedure BUILD
(KB        : in out KB_RECORD;
 FILE_NAME : in     STANDARD.STRING) is
-------------------------------------------------------------------

  KB_FILE          : TEXT_IO.FILE_TYPE;
  CLAUSE           : INPUT_CLAUSE_RECORD;
  TOKEN            : SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.STRING;
  OP               : SYSTEM_TYPES_PACKAGE.
```

```
                    DYNAMIC_STRING.STRING;
  NEW_STRING      : SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.STRING;
  VARIABLES       : VARIABLES_QUEUE.LIST_TYPE;
  ROOT_PTR        : KB_NODE_PTR_TYPE;
  TEMP_PTR        : KB_NODE_PTR_TYPE;
  CLAUSE_HEAD     : KB_NODE_PTR_TYPE;
  ARITY           : NATURAL := 0;
  VARIABLE_COUNT  : VARIABLE_RANGE := 0;

  SYNTAX_ERROR : exception;

-------------------------------------------------------------------
begin

  TEXT_IO.OPEN( KB_FILE,
                TEXT_IO.IN_FILE,
                FILE_NAME);

  -- For each clause in the kb

  while not TEXT_IO.END_OF_FILE(KB_FILE)
  loop

    -- Form head of current predicate

    GET(CLAUSE, KB_FILE);
    GET(TOKEN, CLAUSE);
    ARITY := 0;
    FORM(CLAUSE_HEAD, TOKEN);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_STRING => CLAUSE.SHORT_VALUE,
     TO_THE_STRING   => CLAUSE_HEAD.CLAUSE);

    TEMP_PTR := CLAUSE_HEAD;
    ADD_TO_BACK(CLAUSE_HEAD, KB);
    GET(OP, CLAUSE);


    -- For each token in the clause

    while not SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.
              IS_EQUAL(OP, ".")
    loop

      GET(TOKEN, CLAUSE);

      if SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.IS_EQUAL(OP, ":-") then

        -- A rule therefore reset pointer to head of clause
        -- so that subgoals
        -- can be added

        TEMP_PTR := CLAUSE_HEAD;
        ADD_FIRST_GOAL(TOKEN, TEMP_PTR);
```

```
  ROOT_PTR := TEMP_PTR;
  ARITY := 0;

  CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

elsif SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.IS_EQUAL(OP, ",") then

  ADD_GOAL(TOKEN, TEMP_PTR);

  CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

elsif SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.IS_EQUAL(OP, ";") then

  ADD_OR(TOKEN, ROOT_PTR);
  TEMP_PTR := ROOT_PTR;

 CHECK_IF_VARIABLE( TEMP_PTR, VARIABLES, VARIABLE_COUNT);

elsif SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.IS_EQUAL(OP, "(") then

  FORM_STRUCTURE(TEMP_PTR,
                 TOKEN,
                 CLAUSE,
                 ARITY,
                 VARIABLES,
                 VARIABLE_COUNT);

  -- Set arity of this predicate
  -- then reset ready for next time around if required

 TEMP_PTR.ARITY := ARITY;
  ARITY := 0;

  -- Could have added this detail to the token e.g
  -- DYNAMIC_STRING.PREPEND
  -- (NATURAL'IMAGE(ARITY), TEMP_PTR.NAME);

else

  raise SYNTAX_ERROR;

end if;

GET(OP, CLAUSE);

-- Is this an infix operation?

if SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.IS_EQUAL(OP, "\=") or else
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.IS_EQUAL(OP, "is") or else
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.IS_EQUAL(OP, "<")then

  declare
```

```
        PARAMETER_PTR : KB_NODE_PTR_TYPE;

      begin

        PREFIX_OPERATION(TEMP_PTR,
                         OP,
                         PARAMETER_PTR);

        GET(TOKEN, CLAUSE);
        ADD_GOAL(TOKEN, PARAMETER_PTR);
        CHECK_IF_VARIABLE
        (PARAMETER_PTR, VARIABLES, VARIABLE_COUNT);
        GET(OP, CLAUSE);
        if SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.IS_EQUAL(OP, "(") then
          GET(TOKEN, CLAUSE);
          FORM_STRUCTURE(PARAMETER_PTR,
                         TOKEN,
                         CLAUSE,
                         ARITY,
                         VARIABLES,
                         VARIABLE_COUNT);
          PARAMETER_PTR.ARITY := ARITY;
          ARITY := 0;
          GET(OP, CLAUSE);
        end if;
      end;
    end if;

  end loop;

  -- Transfer variables to clause variable template in clause
  -- head

  if VARIABLE_COUNT /= 0 then

    -- Set discriminant to correct size

    CLAUSE_HEAD.INSTANCE_TEMPLATE :=
    (SIZE           => VARIABLE_COUNT,
     VARIABLES      => (1 .. VARIABLE_COUNT => ( NEW_STRING,
                                                 null,
                                                 null)),
     VARIABLE_COUNT => VARIABLE_COUNT,
     LEVEL          => 0);

    -- transfer the variable details

    for COUNT in 1 .. VARIABLE_COUNT
    loop
      CLAUSE_HEAD.INSTANCE_TEMPLATE.VARIABLES(COUNT).TOKEN :=
      VARIABLES_QUEUE.FRONT_OF(VARIABLES);
      VARIABLES_QUEUE.DELETE_FRONT_OF(VARIABLES);
    end loop;

    CLAUSE_HEAD.INSTANCE_TEMPLATE.VARIABLE_COUNT :=
    VARIABLE_COUNT;

  end if;
```

```
      -- Clear the variable list details ready for next clause

      VARIABLES_QUEUE.CLEAR(VARIABLES);
      VARIABLE_COUNT := 0;

    end loop;

  TEXT_IO.CLOSE(KB_FILE);
-------------------------------------------------------------------
exception
  when SYNTAX_ERROR =>
    TEXT_IO.PUT_LINE
    ("SYNTAX_ERROR raised in LOGIC_KS_PACKAGE.BUILD");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_KB_PACKAGE at " &
     "BUILD");
-------------------------------------------------------------------
end BUILD;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure TEST is
-------------------------------------------------------------------

  -- Test the package
  -- KB.TXT and QUERY.TXT files needed

  package INTEGER_TEXT_IO is new TEXT_IO.INTEGER_IO(INTEGER);
  use INTEGER_TEXT_IO;

  KB     : KB_RECORD;
  QUERY  : KB_RECORD;
  CLAUSE : KB_NODE_PTR_TYPE;

-------------------------------------------------------------------
begin

  TEXT_IO.PUT_LINE("Building Knowledge-Base");
  BUILD(KB, "testkb.txt");
  TEXT_IO.PUT_LINE("Finished");
  CLAUSE := KB.FIRST;

  while CLAUSE /= null
  loop
    PRINT_CLAUSE(CLAUSE);
    TEXT_IO.PUT( '.');
    TEXT_IO.NEW_LINE;
    INTEGER_TEXT_IO.PUT
    (CLAUSE.INSTANCE_TEMPLATE.VARIABLE_COUNT, 2);
    TEXT_IO.PUT(" Variables: ");

    if CLAUSE.INSTANCE_TEMPLATE.VARIABLE_COUNT /= 0 then

      for COUNT in 1 .. CLAUSE.INSTANCE_TEMPLATE.VARIABLE_COUNT
      loop

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.PUT
```

```
            (CLAUSE.INSTANCE_TEMPLATE.VARIABLES( COUNT).TOKEN);
            INTEGER_TEXT_IO.PUT(COUNT, 1);
            TEXT_IO.PUT(" ");

          end loop;

        end if;

        TEXT_IO.NEW_LINE;
        CLAUSE := CLAUSE.NEXT_CLAUSE;

      end loop;

      ASSERT(IN_CLAUSE  => "predecessor(X, pat).",
             KB         =>  QUERY);

      TEXT_IO.PUT_LINE(" The Query is :");
      CLAUSE := QUERY.FIRST;

      while CLAUSE /= null
      loop
         PRINT_CLAUSE(CLAUSE);
         TEXT_IO.PUT('.');
         TEXT_IO.NEW_LINE;
         CLAUSE := CLAUSE.NEXT_CLAUSE;
      end loop;

    ------------------------------------------------------------------
    end TEST;
    ------------------------------------------------------------------


------------------------------------------------------------------------
end LOGIC_KB_PACKAGE;
------------------------------------------------------------------------
```

Logic Inference Engine

```
-------------------------------------------------------------------
--
-- Unit      : LOGIC_INFERENCE_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS Shrivenham
-- Date      : 24 July 1991
-- Function  : This package provides the operations to inference over
--             the knowledge-base produced by Logic_kb_package.
--
-------------------------------------------------------------------
with LOGIC_KB_PACKAGE,
     SYSTEM_TYPES_PACKAGE;
-------------------------------------------------------------------
generic
  NAME : in STANDARD.
            STRING;
-------------------------------------------------------------------
package LOGIC_INFERENCE_PACKAGE is
-------------------------------------------------------------------
  package LOGIC_KB renames LOGIC_KB_PACKAGE;
  use LOGIC_KB; -- Needed to make KB_NODE_PTR_TYPE null value
                -- visible!!

  task SOLVE is
     entry START(THIS_QUERY : in LOGIC_KB.KB_RECORD;
                 THIS_KB    : in LOGIC_KB.KB_RECORD);
  end SOLVE;

  task CONTROL is
    entry PUT_RESULT(
          IN_LIST  : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE);
    entry GET_RESULT(
          OUT_LIST :    out SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE);
    entry ANY_MORE(MORE : out BOOLEAN);
    entry GET_MORE;
    entry NO_MORE;

  end CONTROL;

  procedure TEST;

-------------------------------------------------------------------
end LOGIC_INFERENCE_PACKAGE;
-------------------------------------------------------------------
```

```
------------------------------------------------------------------------
--
-- Unit      : LOGIC_INFERENCE_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS Shrivenham
-- Date      : 24 July 1991
-- Function  : This package provides the operations to inference over
--             the knowledge-base produced by Logic_kb_package.
--
------------------------------------------------------------------------
with TEXT_IO,
     GENERIC_LIST_PACKAGE,
     SYSTEM_TYPES_PACKAGE;
------------------------------------------------------------------------
package body LOGIC_INFERENCE_PACKAGE is
------------------------------------------------------------------------
  -- Package globals
  -- Kb.              The Knowledge-base
  -- Query_variables. A pointer to the query variables

  QUERY_VARIABLES : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;

  -- Instance.  A pointer to the instance variables of the matched
  --            clause
  -- Structure. A pointer to the matched knowledge-base structure

  package INTEGER_TEXT_IO is new TEXT_IO.INTEGER_IO(INTEGER);

  type MATCHED_RECORD is
    record
      INSTANCE  : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
      STRUCTURE : LOGIC_KB.KB_NODE_PTR_TYPE;
    end record;

  -- Instance.  A pointer to the instance variables of the goal
  -- Structure. A pointer to the goal knowledge-base structure

  type GOAL_RECORD is
  record
    INSTANCE  : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
    STRUCTURE : LOGIC_KB.KB_NODE_PTR_TYPE;
  end record;

  -- Instance_record encapsulates a goal and its match

  type INSTANCE_RECORD is
    record
      MATCH : MATCHED_RECORD;
      GOAL  : GOAL_RECORD;
    end record;

  type INSTANCE_RECORD_PTR is access INSTANCE_RECORD;

  -- Bind record is used to record bindings during the unification
  -- process.
  -- Each binding is placed on a queue.  Should the unification fail
  -- or backtrack occur then the queue is used to unbind the
  -- appropriate bindings

  type BIND_RECORD is
```

```
record
  INDEX     : POSITIVE;
  INSTANCE  : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
end record;

type BIND_RECORD_PTR is access BIND_RECORD;

-- Dummy parameters for bind_package instantiation

-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(LEFT  : in BIND_RECORD_PTR;
 RIGHT : in BIND_RECORD_PTR) return BOOLEAN is
-------------------------------------------------------------------
begin
  return TRUE;
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function COPY
-------------------------------------------------------------------
(FROM : in BIND_RECORD_PTR) return BIND_RECORD_PTR is
-------------------------------------------------------------------
begin
  return FROM;
-------------------------------------------------------------------
end COPY;
-------------------------------------------------------------------


-------------------------------------------------------------------
package BIND_PACKAGE is new GENERIC_LIST_PACKAGE
                            (BIND_RECORD_PTR,
                             IS_EQUAL);
-------------------------------------------------------------------

-- Goals provide the operations for manipulating Goals_to_solve

-- Dummy parameters for goals instantiation

-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(LEFT  : in INSTANCE_RECORD_PTR;
 RIGHT : in INSTANCE_RECORD_PTR) return BOOLEAN is
-------------------------------------------------------------------
begin
  return TRUE;
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function COPY
-------------------------------------------------------------------
```

```
(FROM : in INSTANCE_RECORD_PTR) return INSTANCE_RECORD_PTR is
-------------------------------------------------------------------
begin
  return FROM;
-------------------------------------------------------------------
end COPY;
-------------------------------------------------------------------


-------------------------------------------------------------------
package GOALS is new GENERIC_LIST_PACKAGE
                     (INSTANCE_RECORD_PTR,
                      IS_EQUAL);
use GOALS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Put is a diagnostic subprogram which displays the current goal
-- clause and its current matched clause
--
-------------------------------------------------------------------
procedure PUT
-------------------------------------------------------------------
(CURRENT_INSTANCE : in INSTANCE_RECORD_PTR) is
-------------------------------------------------------------------
begin
  TEXT_IO.NEW_LINE;
  TEXT_IO.PUT("Goal  ");
  LOGIC_KB.
  PRINT_PREDICATE
  (PRED => CURRENT_INSTANCE.
           GOAL.
           STRUCTURE);
  TEXT_IO.NEW_LINE;
  TEXT_IO.PUT("Match ");
  LOGIC_KB.
  PRINT_PREDICATE
  (PRED => CURRENT_INSTANCE.
           MATCH.
           STRUCTURE);
  TEXT_IO.NEW_LINE(2);
-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Display_variables is a diagnostic subprogram used to monitor
-- the state of variables as the inferencing process proceedes
--
-------------------------------------------------------------------
procedure DISPLAY_VARIABLES
-------------------------------------------------------------------
(INSTANCE_VARIABLES : in LOGIC_KB.
                         INSTANCE_VARIABLES_RECORD_PTR) is
-------------------------------------------------------------------
begin
```

```
   if (INSTANCE_VARIABLES /= null) then
     for COUNT in 1 .. INSTANCE_VARIABLES.
                       VARIABLE_COUNT
     loop

       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       PUT
       (THE_STRING => INSTANCE_VARIABLES.
                      VARIABLES(COUNT).
                      TOKEN);
       TEXT_IO.PUT(" = ");

       if INSTANCE_VARIABLES.
          VARIABLES(COUNT).
          BOUND_STRUCTURE /= null then

         SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         PUT
         (INSTANCE_VARIABLES.
          VARIABLES(COUNT).
          BOUND_STRUCTURE.
          NAME);

       else

         TEXT_IO.PUT("null");

       end if;

       TEXT_IO.NEW_LINE;

     end loop;

   end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "DISPLAY_VARIABLES");
-------------------------------------------------------------------
end DISPLAY_VARIABLES;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Is_a_variable checks whether an item is a variable
--
-------------------------------------------------------------------
function IS_A_VARIABLE
-------------------------------------------------------------------
(NODE_PTR : in LOGIC_KB.KB_NODE_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
  ITEM : CHARACTER := SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.ITEM_OF(NODE_PTR.NAME, 1);
-------------------------------------------------------------------
```

```
  begin

    return (ITEM in 'A' .. 'Z') or (ITEM ='_');

  ------------------------------------------------------------------
  end IS_A_VARIABLE;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  --
  -- Find follows the binding pointers in order to establish the
  -- bound structure and its variables
  --
  ------------------------------------------------------------------
  procedure FIND
  ------------------------------------------------------------------
  (INSTANTIATED_STRUCTURE_PTR :    out LOGIC_KB.KB_NODE_PTR_TYPE;
   INSTANTIATED_VARIABLES_PTR :    out LOGIC_KB.
                                       INSTANCE_VARIABLES_RECORD_PTR;
   STRUCTURE_PTR               : in    LOGIC_KB.KB_NODE_PTR_TYPE;
   VARIABLES_PTR               : in    LOGIC_KB.
                                       INSTANCE_VARIABLES_RECORD_PTR)
  is
  ------------------------------------------------------------------
    TEMP_STRUCTURE_PTR : LOGIC_KB.KB_NODE_PTR_TYPE;
    TEMP_VARIABLES_PTR : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
    INDEX              : NATURAL;
  ------------------------------------------------------------------
  begin

    if VARIABLES_PTR = null then

      -- Instantiated structure has no variables

      INSTANTIATED_STRUCTURE_PTR := STRUCTURE_PTR;

      INSTANTIATED_VARIABLES_PTR := null;

    else

      -- Instantiated structure has variables

      INDEX := STRUCTURE_PTR.INDEX;
      TEMP_STRUCTURE_PTR := VARIABLES_PTR.
                            VARIABLES(INDEX).BOUND_STRUCTURE;

      TEMP_VARIABLES_PTR := VARIABLES_PTR.
                            VARIABLES(INDEX).BOUND_VALUE;

      while (IS_A_VARIABLE(TEMP_STRUCTURE_PTR)) and then
            (TEMP_VARIABLES_PTR.
             VARIABLES(TEMP_STRUCTURE_PTR.INDEX).
             BOUND_STRUCTURE /= null)
      loop

        INDEX := TEMP_STRUCTURE_PTR.INDEX;
        TEMP_STRUCTURE_PTR := TEMP_VARIABLES_PTR.
                              VARIABLES(INDEX).BOUND_STRUCTURE;
```

```
      if TEMP_VARIABLES_PTR.
         VARIABLES(INDEX).BOUND_VALUE /= null then

        -- Has clause has variables

        TEMP_VARIABLES_PTR :=
          TEMP_VARIABLES_PTR.VARIABLES(INDEX).BOUND_VALUE;

      end if;

    end loop;

    INSTANTIATED_STRUCTURE_PTR := TEMP_STRUCTURE_PTR;
    INSTANTIATED_VARIABLES_PTR := TEMP_VARIABLES_PTR;

   end if;

-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "FIND");
-------------------------------------------------------------------
end FIND;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Display_solution finds the bindings for each query variable and
-- displays them
--
-------------------------------------------------------------------
procedure DISPLAY_SOLUTION
-------------------------------------------------------------------
(QUERY_VARIABLES_PTR : in     LOGIC_KB.
                              INSTANCE_VARIABLES_RECORD_PTR) is
-------------------------------------------------------------------

  INSTANTIATED_STRUCTURE_PTR : LOGIC_KB.KB_NODE_PTR_TYPE;
  INSTANTIATED_VARIABLES_PTR : LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;
-------------------------------------------------------------------
begin

  if QUERY_VARIABLES_PTR /= null then
    for COUNT in 1 .. QUERY_VARIABLES_PTR.VARIABLE_COUNT
    loop

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => VALUE_LIST,
       THIS_ITEM => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
```

```
                    COPY
                    (FROM_THE_STRING => QUERY_VARIABLES_PTR.
                                        VARIABLES(COUNT).TOKEN));

        FIND(INSTANTIATED_STRUCTURE_PTR,
             INSTANTIATED_VARIABLES_PTR,
             QUERY_VARIABLES.VARIABLES(COUNT).BOUND_STRUCTURE,
             QUERY_VARIABLES.VARIABLES(COUNT).BOUND_VALUE);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => VALUE_LIST,
       THIS_ITEM => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    COPY
                    (FROM_THE_STRING =>
                     INSTANTIATED_STRUCTURE_PTR.NAME));

    end loop;

  else

    declare

      YES : SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            STRING;
    begin

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_SUBSTRING => "Yes",
       TO_THE_STRING      => YES);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => VALUE_LIST,
       THIS_ITEM => YES);

    end;

  end if;

  CONTROL.
  PUT_RESULT
  (IN_LIST => VALUE_LIST);
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "DISPLAY_SOLUTION");
-------------------------------------------------------------------
end DISPLAY_SOLUTION;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Is_a_rule checks whether the matched structure is a rule.
--
-------------------------------------------------------------------
function IS_A_RULE
-------------------------------------------------------------------
(NODE_PTR : in LOGIC_KB.KB_NODE_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin

  return (NODE_PTR.NEXT_GOAL /= null);

-------------------------------------------------------------------
end IS_A_RULE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Bind_elements binds the goal and matched elements.
-- Bindings are recorded on the Bind_queue
--
-------------------------------------------------------------------
procedure BIND_ELEMENTS
-------------------------------------------------------------------
(GOAL_PTR             : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 GOAL_VARIABLES_PTR   : in     LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR;
 MATCH_PTR            : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 MATCH_VARIABLES_PTR  : in     LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR;
 BIND_QUEUE           : in out BIND_PACKAGE.LIST_TYPE;
 UNIFIED              : in out BOOLEAN) is
-------------------------------------------------------------------
  BOUND_ELEMENT_PTR    : BIND_RECORD_PTR;
-------------------------------------------------------------------
begin

  if ( not IS_A_VARIABLE(GOAL_PTR)) and then
     ( not IS_A_VARIABLE(MATCH_PTR)) then

    -- Both constants

    if (not SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.IS_EQUAL
                          (GOAL_PTR.NAME, MATCH_PTR.NAME)) then

       UNIFIED := FALSE;

    end if;


  elsif (not IS_A_VARIABLE(GOAL_PTR)) and then
        (IS_A_VARIABLE(MATCH_PTR)) then

    -- Goal is a constant and the match is a variable
    -- Bind match to goal

    MATCH_VARIABLES_PTR.
```

```
      VARIABLES(MATCH_PTR.INDEX).BOUND_STRUCTURE := GOAL_PTR;
      MATCH_VARIABLES_PTR.
      VARIABLES(MATCH_PTR.INDEX).BOUND_VALUE := GOAL_VARIABLES_PTR;

      -- Record binding

      BOUND_ELEMENT_PTR := new BIND_RECORD;
      BOUND_ELEMENT_PTR.INSTANCE := MATCH_VARIABLES_PTR;
      BOUND_ELEMENT_PTR.INDEX := MATCH_PTR.INDEX;
      BIND_PACKAGE.PUT_ON_BACK_OF(BIND_QUEUE, BOUND_ELEMENT_PTR);

    else

      -- Goal is a variable and the match is either a variable or a
      -- constant

      if (GOAL_VARIABLES_PTR = MATCH_VARIABLES_PTR) and then
         (GOAL_PTR.INDEX = MATCH_PTR.INDEX) then

        null;

        -- Do not bind a variable to itself otherwise an infinite
        -- loop
        -- is created when Find tries to locate the binding

      else

        -- Both are variables.   Bind goal to match

        GOAL_VARIABLES_PTR.
        VARIABLES(GOAL_PTR.INDEX).BOUND_STRUCTURE := MATCH_PTR;
        GOAL_VARIABLES_PTR.
        VARIABLES(GOAL_PTR.INDEX).
        BOUND_VALUE := MATCH_VARIABLES_PTR;

        -- Record binding

        BOUND_ELEMENT_PTR := new BIND_RECORD;
        BOUND_ELEMENT_PTR.INSTANCE := GOAL_VARIABLES_PTR;
        BOUND_ELEMENT_PTR.INDEX := GOAL_PTR.INDEX;
        BIND_PACKAGE.PUT_ON_BACK_OF(BIND_QUEUE, BOUND_ELEMENT_PTR);

      end if;

    end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "BIND_ELEMENTS");
-------------------------------------------------------------------
end BIND_ELEMENTS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Unbind uses the detail recorded on the Bind_queue to unbind
-- elements
```

```
--
-------------------------------------------------------------------
procedure UNBIND
-------------------------------------------------------------------
(BIND_QUEUE : in out BIND_PACKAGE.LIST_TYPE) is
-------------------------------------------------------------------
  BOUND_ELEMENT_PTR : BIND_RECORD_PTR;
-------------------------------------------------------------------
begin

  while not BIND_PACKAGE.IS_EMPTY(BIND_QUEUE)
  loop

    -- Get binding

    BIND_PACKAGE.GET_FROM_FRONT_OF(BIND_QUEUE, BOUND_ELEMENT_PTR);

    -- Unbind

    BOUND_ELEMENT_PTR.INSTANCE.VARIABLES(BOUND_ELEMENT_PTR.INDEX)
                                    .BOUND_STRUCTURE := null;
    BOUND_ELEMENT_PTR.INSTANCE.VARIABLES(BOUND_ELEMENT_PTR.INDEX)
                                    .BOUND_VALUE := null;

  end loop;

-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "UNBIND");

-------------------------------------------------------------------
end UNBIND;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Is_bound checks to whether a particular variable is currently
-- bound
--
-------------------------------------------------------------------
function IS_BOUND
-------------------------------------------------------------------
(GOAL_PTR           : in LOGIC_KB.KB_NODE_PTR_TYPE;
 GOAL_VARIABLES_PTR : in LOGIC_KB.
                         INSTANCE_VARIABLES_RECORD_PTR)
return BOOLEAN is
-------------------------------------------------------------------
begin

  return (GOAL_VARIABLES_PTR.
          VARIABLES
          (GOAL_PTR.INDEX).BOUND_STRUCTURE /= null);

-------------------------------------------------------------------
end IS_BOUND;
-------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unify_parameters is a recursive subprogram which attempts to
-- unify each of the parameters of the current goal and matched
-- structures
--
------------------------------------------------------------------
procedure UNIFY_PARAMETERS
------------------------------------------------------------------
(GOAL_PTR             : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 GOAL_VARIABLES_PTR   : in     LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR;
 MATCH_PTR            : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 MATCH_VARIABLES_PTR  : in     LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR;
 BIND_QUEUE           : in out BIND_PACKAGE.LIST_TYPE;
 UNIFIED              : in out BOOLEAN) is
------------------------------------------------------------------
  CURRENT_GOAL_PTR         : LOGIC_KB.
                             KB_NODE_PTR_TYPE := GOAL_PTR;
  CURRENT_MATCH_PTR        : LOGIC_KB.
                             KB_NODE_PTR_TYPE := MATCH_PTR;
  TEMP_GOAL_PTR            : LOGIC_KB.
                             KB_NODE_PTR_TYPE := GOAL_PTR;
  TEMP_GOAL_VARIABLES_PTR  : LOGIC_KB.
                             INSTANCE_VARIABLES_RECORD_PTR :=
                             GOAL_VARIABLES_PTR;
  TEMP_MATCH_PTR           : LOGIC_KB.
                             KB_NODE_PTR_TYPE := MATCH_PTR;
  TEMP_MATCH_VARIABLES_PTR : LOGIC_KB.
                             INSTANCE_VARIABLES_RECORD_PTR :=
                             MATCH_VARIABLES_PTR;

  GOAL_IS_VAR    : BOOLEAN;
  MATCH_IS_VAR   : BOOLEAN;
  GOAL_IS_BOUND  : BOOLEAN;
  MATCH_IS_BOUND : BOOLEAN;
------------------------------------------------------------------
begin

  -- Move current_goal_ptr and current_match_ptr along current
  -- goal and matched clauses

  while (CURRENT_GOAL_PTR /= null) and (CURRENT_MATCH_PTR /= null)
  loop

    GOAL_IS_VAR := IS_A_VARIABLE(CURRENT_GOAL_PTR);
    MATCH_IS_VAR := IS_A_VARIABLE(CURRENT_MATCH_PTR);

    if (not GOAL_IS_VAR) and then (not MATCH_IS_VAR) then

      -- Both are constants

      BIND_ELEMENTS(CURRENT_GOAL_PTR,
                    GOAL_VARIABLES_PTR,
                    CURRENT_MATCH_PTR,
                    MATCH_VARIABLES_PTR,
                    BIND_QUEUE,
                    UNIFIED);
```

```
-- Check for parameters at this point in the clause

if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
    (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

  UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                   GOAL_VARIABLES_PTR,
                   CURRENT_MATCH_PTR.NEXT_PARAMETER,
                   MATCH_VARIABLES_PTR,
                   BIND_QUEUE,
                   UNIFIED);

end if;

elsif (not GOAL_IS_VAR) and then (MATCH_IS_VAR) then

  -- Goal is a constant and match is a variable

  if IS_BOUND(CURRENT_MATCH_PTR, MATCH_VARIABLES_PTR) then

    -- Find match instantiation

    FIND(TEMP_MATCH_PTR,
         TEMP_MATCH_VARIABLES_PTR,
         CURRENT_MATCH_PTR,
         MATCH_VARIABLES_PTR);

    BIND_ELEMENTS(CURRENT_GOAL_PTR,
                  GOAL_VARIABLES_PTR,
                  TEMP_MATCH_PTR,
                  TEMP_MATCH_VARIABLES_PTR,
                  BIND_QUEUE,
                  UNIFIED);

    -- Check for parameters at this point in the clause

    if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
       (TEMP_MATCH_PTR.NEXT_PARAMETER /= null) then

      UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                       GOAL_VARIABLES_PTR,
                       TEMP_MATCH_PTR.NEXT_PARAMETER,
                       TEMP_MATCH_VARIABLES_PTR,
                       BIND_QUEUE,
                       UNIFIED);

    end if;

  else

    -- Match variable not instantiated

    BIND_ELEMENTS(CURRENT_GOAL_PTR,
                  GOAL_VARIABLES_PTR,
                  CURRENT_MATCH_PTR,
                  MATCH_VARIABLES_PTR,
                  BIND_QUEUE,
                  UNIFIED);

    -- Check for parameters at this point in the clause
```

```
      if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
         (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

        UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                         GOAL_VARIABLES_PTR,
                         CURRENT_MATCH_PTR.NEXT_PARAMETER,
                         MATCH_VARIABLES_PTR,
                         BIND_QUEUE,
                         UNIFIED);

      end if;


    end if;

  elsif (GOAL_IS_VAR) and then (not MATCH_IS_VAR) then

    -- Goal is a variable and match is a constant

    if IS_BOUND(CURRENT_GOAL_PTR, GOAL_VARIABLES_PTR) then

      -- Then find goal variable instantiation

      FIND(TEMP_GOAL_PTR,
           TEMP_GOAL_VARIABLES_PTR,
           CURRENT_GOAL_PTR,
           GOAL_VARIABLES_PTR);

      BIND_ELEMENTS(TEMP_GOAL_PTR,
                    TEMP_GOAL_VARIABLES_PTR,
                    CURRENT_MATCH_PTR,
                    MATCH_VARIABLES_PTR,
                    BIND_QUEUE,
                    UNIFIED);

      -- Check for parameters at this point in the clause

      if (TEMP_GOAL_PTR.NEXT_PARAMETER /= null) and then
         (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

        UNIFY_PARAMETERS(TEMP_GOAL_PTR.NEXT_PARAMETER,
                         TEMP_GOAL_VARIABLES_PTR,
                         CURRENT_MATCH_PTR.NEXT_PARAMETER,
                         MATCH_VARIABLES_PTR,
                         BIND_QUEUE,
                         UNIFIED);

      end if;

    else

      -- Goal variable not bound

      BIND_ELEMENTS(CURRENT_GOAL_PTR,
                    GOAL_VARIABLES_PTR,
                    CURRENT_MATCH_PTR,
                    MATCH_VARIABLES_PTR,
                    BIND_QUEUE,
                    UNIFIED);
```

```
      -- Check for parameters at this point in the clause

      if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
         (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

        UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                         GOAL_VARIABLES_PTR,
                         CURRENT_MATCH_PTR.NEXT_PARAMETER,
                         MATCH_VARIABLES_PTR,
                         BIND_QUEUE,
                         UNIFIED);

      end if;

    end if;

  else

      -- Both are variables

      GOAL_IS_BOUND := IS_BOUND
                       (CURRENT_GOAL_PTR, GOAL_VARIABLES_PTR);
      MATCH_IS_BOUND := IS_BOUND
                        (CURRENT_MATCH_PTR, MATCH_VARIABLES_PTR);

      if (not GOAL_IS_BOUND) and then (not MATCH_IS_BOUND) then

        -- Neither are bound.   Bind gaol to match

        BIND_ELEMENTS(CURRENT_GOAL_PTR,
                      GOAL_VARIABLES_PTR,
                      CURRENT_MATCH_PTR,
                      MATCH_VARIABLES_PTR,
                      BIND_QUEUE,
                      UNIFIED);

        -- Check for parameters at this point in the clause

        if CURRENT_GOAL_PTR.NEXT_PARAMETER /= null then

          UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                           GOAL_VARIABLES_PTR,
                           CURRENT_MATCH_PTR.NEXT_PARAMETER,
                           MATCH_VARIABLES_PTR,
                           BIND_QUEUE,
                           UNIFIED);

        end if;

      elsif (not GOAL_IS_BOUND) and then (MATCH_IS_BOUND) then

        -- Bind goal to match

        FIND(TEMP_MATCH_PTR,
             TEMP_MATCH_VARIABLES_PTR,
             CURRENT_MATCH_PTR,
             MATCH_VARIABLES_PTR);

        BIND_ELEMENTS(CURRENT_GOAL_PTR,
```

```
                    GOAL_VARIABLES_PTR,
                    TEMP_MATCH_PTR,
                    TEMP_MATCH_VARIABLES_PTR,
                    BIND_QUEUE,
                    UNIFIED);

    -- Check for parameters at this point in the clause

   if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
      (TEMP_MATCH_PTR.NEXT_PARAMETER /= null) then

     UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                      GOAL_VARIABLES_PTR,
                      TEMP_MATCH_PTR.NEXT_PARAMETER,
                      TEMP_MATCH_VARIABLES_PTR,
                      BIND_QUEUE,
                      UNIFIED);

   end if;

elsif (GOAL_IS_BOUND) and then (not MATCH_IS_BOUND) then

   -- Bind match to goal

   FIND(TEMP_GOAL_PTR,
        TEMP_GOAL_VARIABLES_PTR,
        CURRENT_GOAL_PTR,
        GOAL_VARIABLES_PTR);

   BIND_ELEMENTS(TEMP_GOAL_PTR,
                 TEMP_GOAL_VARIABLES_PTR,
                 CURRENT_MATCH_PTR,
                 MATCH_VARIABLES_PTR,
                 BIND_QUEUE,
                 UNIFIED);

   -- Check for parameters at this point in the query

   if (TEMP_GOAL_PTR.NEXT_PARAMETER /= null) and then
      (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

     UNIFY_PARAMETERS(TEMP_GOAL_PTR.NEXT_PARAMETER,
                      TEMP_GOAL_VARIABLES_PTR,
                      CURRENT_MATCH_PTR.NEXT_PARAMETER,
                      MATCH_VARIABLES_PTR,
                      BIND_QUEUE,
                      UNIFIED);

   end if;


elsif (GOAL_IS_BOUND) and then (MATCH_IS_BOUND) then

    -- Find instantiations and check for a match

    FIND(TEMP_GOAL_PTR,
         TEMP_GOAL_VARIABLES_PTR,
         CURRENT_GOAL_PTR,
         GOAL_VARIABLES_PTR);
```

```
            FIND(TEMP_MATCH_PTR,
                 TEMP_MATCH_VARIABLES_PTR,
                 CURRENT_MATCH_PTR,
                 MATCH_VARIABLES_PTR);

            BIND_ELEMENTS(TEMP_GOAL_PTR,
                          TEMP_GOAL_VARIABLES_PTR,
                          TEMP_MATCH_PTR,
                          TEMP_MATCH_VARIABLES_PTR,
                          BIND_QUEUE,
                          UNIFIED);

            -- Check for parameters at this point

            if (TEMP_GOAL_PTR.NEXT_PARAMETER /= null) and then
               (TEMP_MATCH_PTR.NEXT_PARAMETER /= null) then

              UNIFY_PARAMETERS(TEMP_GOAL_PTR.NEXT_PARAMETER,
                               TEMP_GOAL_VARIABLES_PTR,
                               TEMP_MATCH_PTR.NEXT_PARAMETER,
                               TEMP_MATCH_VARIABLES_PTR,
                               BIND_QUEUE,
                               UNIFIED);

            end if;
          end if;
        end if;

        exit when not UNIFIED;

        -- Move along the clause to the next element

        CURRENT_GOAL_PTR := CURRENT_GOAL_PTR.NEXT_GOAL;
        CURRENT_MATCH_PTR := CURRENT_MATCH_PTR.NEXT_GOAL;

      end loop;

-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "UNIFY_PARAMETERS");
-------------------------------------------------------------------
end UNIFY_PARAMETERS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Unify_operands attempts to unify the operands of a built-in
-- operation
--
-------------------------------------------------------------------
procedure UNIFY_OPERANDS
-------------------------------------------------------------------
(GOAL_PTR              : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 GOAL_VARIABLES_PTR    : in     LOGIC_KB.
                                INSTANCE_VARIABLES_RECORD_PTR;
 MATCH_PTR             : in     LOGIC_KB.KB_NODE_PTR_TYPE;
```

```
  MATCH_VARIABLES_PTR : in     LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR;
  BIND_QUEUE          : in out BIND_PACKAGE.LIST_TYPE;
  UNIFIED             : in out BOOLEAN) is
-------------------------------------------------------------------
   CURRENT_GOAL_PTR          : LOGIC_KB.
                               KB_NODE_PTR_TYPE := GOAL_PTR;
   CURRENT_MATCH_PTR         : LOGIC_KB.KB_NODE_PTR_TYPE :=
                               MATCH_PTR;
   TEMP_GOAL_PTR             : LOGIC_KB.
                               KB_NODE_PTR_TYPE := GOAL_PTR;
   TEMP_GOAL_VARIABLES_PTR   : LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR :=
                               GOAL_VARIABLES_PTR;
   TEMP_MATCH_PTR            : LOGIC_KB.KB_NODE_PTR_TYPE :=
                               MATCH_PTR;
   TEMP_MATCH_VARIABLES_PTR  : LOGIC_KB.
                               INSTANCE_VARIABLES_RECORD_PTR :=
                               MATCH_VARIABLES_PTR;

   GOAL_IS_VAR     : BOOLEAN;
   MATCH_IS_VAR    : BOOLEAN;
   GOAL_IS_BOUND   : BOOLEAN;
   MATCH_IS_BOUND  : BOOLEAN;
-------------------------------------------------------------------
begin

    GOAL_IS_VAR := IS_A_VARIABLE(CURRENT_GOAL_PTR);
    MATCH_IS_VAR := IS_A_VARIABLE(CURRENT_MATCH_PTR);

    if (not GOAL_IS_VAR) and then (not MATCH_IS_VAR) then

      -- Both are constants

      BIND_ELEMENTS(CURRENT_GOAL_PTR,
                    GOAL_VARIABLES_PTR,
                    CURRENT_MATCH_PTR,
                    MATCH_VARIABLES_PTR,
                    BIND_QUEUE,
                    UNIFIED);

     -- Check for parameters at this point in the clause

     if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
         (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

       UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                        GOAL_VARIABLES_PTR,
                        CURRENT_MATCH_PTR.NEXT_PARAMETER,
                        MATCH_VARIABLES_PTR,
                        BIND_QUEUE,
                        UNIFIED);

     end if;

    elsif (not GOAL_IS_VAR) and then (MATCH_IS_VAR) then

      -- Goal is a constant and match is a variable

      if IS_BOUND(CURRENT_MATCH_PTR, MATCH_VARIABLES_PTR) then
```

```
    -- Find match instantiation

    FIND(TEMP_MATCH_PTR,
         TEMP_MATCH_VARIABLES_PTR,
         CURRENT_MATCH_PTR,
         MATCH_VARIABLES_PTR);

    BIND_ELEMENTS(CURRENT_GOAL_PTR,
                  GOAL_VARIABLES_PTR,
                  TEMP_MATCH_PTR,
                  TEMP_MATCH_VARIABLES_PTR,
                  BIND_QUEUE,
                  UNIFIED);

    -- Check for parameters at this point in the clause

    if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
       (TEMP_MATCH_PTR.NEXT_PARAMETER /= null) then

      UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                       GOAL_VARIABLES_PTR,
                       TEMP_MATCH_PTR.NEXT_PARAMETER,
                       TEMP_MATCH_VARIABLES_PTR,
                       BIND_QUEUE,
                       UNIFIED);

    end if;

  else

    -- Match variable not instantiated

    BIND_ELEMENTS(CURRENT_GOAL_PTR,
                  GOAL_VARIABLES_PTR,
                  CURRENT_MATCH_PTR,
                  MATCH_VARIABLES_PTR,
                  BIND_QUEUE,
                  UNIFIED);

    -- Check for parameters at this point in the clause

    if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
       (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

      UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                       GOAL_VARIABLES_PTR,
                       CURRENT_MATCH_PTR.NEXT_PARAMETER,
                       MATCH_VARIABLES_PTR,
                       BIND_QUEUE,
                       UNIFIED);

    end if;

  end if;

elsif (GOAL_IS_VAR) and then (not MATCH_IS_VAR) then

  -- Goal is a variable and match is a constant
```

```
if IS_BOUND(CURRENT_GOAL_PTR, GOAL_VARIABLES_PTR) then

  -- Then find goal variable instantiation

  FIND(TEMP_GOAL_PTR,
       TEMP_GOAL_VARIABLES_PTR,
       CURRENT_GOAL_PTR,
       GOAL_VARIABLES_PTR);

  BIND_ELEMENTS(TEMP_GOAL_PTR,
                TEMP_GOAL_VARIABLES_PTR,
                CURRENT_MATCH_PTR,
                MATCH_VARIABLES_PTR,
                BIND_QUEUE,
                UNIFIED);

  -- Check for parameters at this point in the clause

  if (TEMP_GOAL_PTR.NEXT_PARAMETER /= null) and then
     (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

    UNIFY_PARAMETERS(TEMP_GOAL_PTR.NEXT_PARAMETER,
                     TEMP_GOAL_VARIABLES_PTR,
                     CURRENT_MATCH_PTR.NEXT_PARAMETER,
                     MATCH_VARIABLES_PTR,
                     BIND_QUEUE,
                     UNIFIED);

  end if;

else

  -- Goal variable not bound

  BIND_ELEMENTS(CURRENT_GOAL_PTR,
                GOAL_VARIABLES_PTR,
                CURRENT_MATCH_PTR,
                MATCH_VARIABLES_PTR,
                BIND_QUEUE,
                UNIFIED);

  -- Check for parameters at this point in the clause

  if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
     (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

    UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                     GOAL_VARIABLES_PTR,
                     CURRENT_MATCH_PTR.NEXT_PARAMETER,
                     MATCH_VARIABLES_PTR,
                     BIND_QUEUE,
                     UNIFIED);

  end if;

 end if;

else

  -- Both are variables
```

```
GOAL_IS_BOUND := IS_BOUND
                 (CURRENT_GOAL_PTR, GOAL_VARIABLES_PTR);
MATCH_IS_BOUND := IS_BOUND
                  (CURRENT_MATCH_PTR, MATCH_VARIABLES_PTR);

if (not GOAL_IS_BOUND) and then (not MATCH_IS_BOUND) then

  -- Neither are bound.   Bind gaol to match

  BIND_ELEMENTS(CURRENT_GOAL_PTR,
                GOAL_VARIABLES_PTR,
                CURRENT_MATCH_PTR,
                MATCH_VARIABLES_PTR,
                BIND_QUEUE,
                UNIFIED);

  -- Check for parameters at this point in the clause

  if CURRENT_GOAL_PTR.NEXT_PARAMETER /= null then

    UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                     GOAL_VARIABLES_PTR,
                     CURRENT_MATCH_PTR.NEXT_PARAMETER,
                     MATCH_VARIABLES_PTR,
                     BIND_QUEUE,
                     UNIFIED);

  end if;

elsif (not GOAL_IS_BOUND) and then (MATCH_IS_BOUND) then

  -- Bind goal to match

  FIND(TEMP_MATCH_PTR,
       TEMP_MATCH_VARIABLES_PTR,
       CURRENT_MATCH_PTR,
       MATCH_VARIABLES_PTR);

  BIND_ELEMENTS(CURRENT_GOAL_PTR,
                GOAL_VARIABLES_PTR,
                TEMP_MATCH_PTR,
                TEMP_MATCH_VARIABLES_PTR,
                BIND_QUEUE,
                UNIFIED);

   -- Check for parameters at this point in the clause

  if (CURRENT_GOAL_PTR.NEXT_PARAMETER /= null) and then
     (TEMP_MATCH_PTR.NEXT_PARAMETER /= null) then

    UNIFY_PARAMETERS(CURRENT_GOAL_PTR.NEXT_PARAMETER,
                     GOAL_VARIABLES_PTR,
                     TEMP_MATCH_PTR.NEXT_PARAMETER,
                     TEMP_MATCH_VARIABLES_PTR,
                     BIND_QUEUE,
                     UNIFIED);

  end if;
```

```
elsif (GOAL_IS_BOUND) and then (not MATCH_IS_BOUND) then

  -- Bind match to goal

  FIND(TEMP_GOAL_PTR,
       TEMP_GOAL_VARIABLES_PTR,
       CURRENT_GOAL_PTR,
       GOAL_VARIABLES_PTR);

  BIND_ELEMENTS(TEMP_GOAL_PTR,
                TEMP_GOAL_VARIABLES_PTR,
                CURRENT_MATCH_PTR,
                MATCH_VARIABLES_PTR,
                BIND_QUEUE,
                UNIFIED);

  -- Check for parameters at this point in the query

  if (TEMP_GOAL_PTR.NEXT_PARAMETER /= null) and then
     (CURRENT_MATCH_PTR.NEXT_PARAMETER /= null) then

    UNIFY_PARAMETERS(TEMP_GOAL_PTR.NEXT_PARAMETER,
                     TEMP_GOAL_VARIABLES_PTR,
                     CURRENT_MATCH_PTR.NEXT_PARAMETER,
                     MATCH_VARIABLES_PTR,
                     BIND_QUEUE,
                     UNIFIED);

  end if;

elsif (GOAL_IS_BOUND) and then (MATCH_IS_BOUND) then

   -- Find instantiations and check for a match

   FIND(TEMP_GOAL_PTR,
        TEMP_GOAL_VARIABLES_PTR,
        CURRENT_GOAL_PTR,
        GOAL_VARIABLES_PTR);

   FIND(TEMP_MATCH_PTR,
        TEMP_MATCH_VARIABLES_PTR,
        CURRENT_MATCH_PTR,
        MATCH_VARIABLES_PTR);

   BIND_ELEMENTS(TEMP_GOAL_PTR,
                 TEMP_GOAL_VARIABLES_PTR,
                 TEMP_MATCH_PTR,
                 TEMP_MATCH_VARIABLES_PTR,
                 BIND_QUEUE,
                 UNIFIED);

   -- Check for parameters at this point

   if (TEMP_GOAL_PTR.NEXT_PARAMETER /= null) and then
      (TEMP_MATCH_PTR.NEXT_PARAMETER /= null) then

     UNIFY_PARAMETERS(TEMP_GOAL_PTR.NEXT_PARAMETER,
                      TEMP_GOAL_VARIABLES_PTR,
                      TEMP_MATCH_PTR.NEXT_PARAMETER,
                      TEMP_MATCH_VARIABLES_PTR,
```

```
                                BIND_QUEUE,
                                UNIFIED);

          end if;
        end if;
      end if;

-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "UNIFY_OPERANDS");
-------------------------------------------------------------------
end UNIFY_OPERANDS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Unify attempts to unify the current goal and matched structures
--
-------------------------------------------------------------------
procedure UNIFY
-------------------------------------------------------------------
(GOAL_HEAD_PTR       : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 GOAL_VARIABLES_PTR  : in     LOGIC_KB.
                              INSTANCE_VARIABLES_RECORD_PTR;
 MATCH_HEAD_PTR      : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 MATCH_VARIABLES_PTR : in     LOGIC_KB.
                              INSTANCE_VARIABLES_RECORD_PTR;
 BIND_QUEUE          : in out BIND_PACKAGE.LIST_TYPE;
 UNIFIED             : in out BOOLEAN) is
-------------------------------------------------------------------
begin

  -- Unify head of predicate

  BIND_ELEMENTS(GOAL_HEAD_PTR,
                GOAL_VARIABLES_PTR,
                MATCH_HEAD_PTR,
                MATCH_VARIABLES_PTR,
                BIND_QUEUE,
                UNIFIED);

  if UNIFIED then

    -- Check for parameters at this point in the clause

    if (GOAL_HEAD_PTR.NEXT_PARAMETER /= null) and then
         (MATCH_HEAD_PTR.NEXT_PARAMETER /= null) then

      UNIFY_PARAMETERS(GOAL_HEAD_PTR.NEXT_PARAMETER,
                       GOAL_VARIABLES_PTR,
                       MATCH_HEAD_PTR.NEXT_PARAMETER,
                       MATCH_VARIABLES_PTR,
                       BIND_QUEUE,
                       UNIFIED);
    end if;
  end if;
```

```
------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "UNIFY");
------------------------------------------------------------------
end UNIFY;
------------------------------------------------------------------


------------------------------------------------------------------
--
-- Create builds a copy of the clause variables instance_template
-- located in the head of the current goal structure
--
------------------------------------------------------------------
procedure CREATE
------------------------------------------------------------------
(CURRENT_LEVEL           : in      NATURAL;
 INSTANCE_VARIABLES_PTR :     out LOGIC_KB.
                                  INSTANCE_VARIABLES_RECORD_PTR;
 GOAL_PTR                : in      LOGIC_KB.KB_NODE_PTR_TYPE) is
------------------------------------------------------------------
  VARIABLE_COUNT      : NATURAL :=
                        GOAL_PTR.INSTANCE_TEMPLATE.VARIABLE_COUNT;
  TEMP_VARIABLES_PTR : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
------------------------------------------------------------------
begin

  if VARIABLE_COUNT /= 0 then

    -- Create a variable record for this instance

    TEMP_VARIABLES_PTR := new
                          INSTANCE_VARIABLES_RECORD
                          (VARIABLE_COUNT);

    -- Copy details from template to instance

    for COUNT in 1 .. VARIABLE_COUNT
    loop

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (GOAL_PTR.INSTANCE_TEMPLATE.VARIABLES(COUNT).TOKEN,
       TEMP_VARIABLES_PTR.VARIABLES(COUNT).TOKEN);

    end loop;

    TEMP_VARIABLES_PTR.VARIABLE_COUNT := VARIABLE_COUNT;
    TEMP_VARIABLES_PTR.LEVEL := CURRENT_LEVEL;

    INSTANCE_VARIABLES_PTR := TEMP_VARIABLES_PTR;

  else

    -- There are no variables associated with this clause
```

```
    INSTANCE_VARIABLES_PTR := null;

  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "CREATE_1");
-------------------------------------------------------------------
end CREATE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Create produces a goal instance which will be placed on the
-- goals_to_solve queue
--
-------------------------------------------------------------------
procedure CREATE
-------------------------------------------------------------------
(GOAL_INSTANCE_PTR      :     out INSTANCE_RECORD_PTR;
 GOAL_PTR               : in    LOGIC_KB.KB_NODE_PTR_TYPE;
 INSTANCE_VARIABLES_PTR : in    LOGIC_KB.
                                INSTANCE_VARIABLES_RECORD_PTR)
is
-------------------------------------------------------------------
  TEMP_INSTANCE_PTR : INSTANCE_RECORD_PTR;
-------------------------------------------------------------------
begin

  -- Create a composite goal/match record

  TEMP_INSTANCE_PTR := new INSTANCE_RECORD;
  TEMP_INSTANCE_PTR.GOAL.INSTANCE := INSTANCE_VARIABLES_PTR;
  TEMP_INSTANCE_PTR.GOAL.STRUCTURE := GOAL_PTR;
  GOAL_INSTANCE_PTR := TEMP_INSTANCE_PTR;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "CREATE_2");
-------------------------------------------------------------------
end CREATE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Match tries to find a match for the current goal instance
-- against the  knowledge-base
--
-------------------------------------------------------------------
procedure MATCH
-------------------------------------------------------------------
(KB                     : in    LOGIC_KB.KB_RECORD;
 CURRENT_LEVEL          : in    NATURAL;
 CURRENT_GOAL_INSTANCE  : in    INSTANCE_RECORD_PTR;
```

```
 MATCHED                :    out BOOLEAN) is
-------------------------------------------------------------------
  CLAUSE_PTR    : LOGIC_KB.KB_NODE_PTR_TYPE :=
                  CURRENT_GOAL_INSTANCE.MATCH.STRUCTURE;
  VARIABLES_PTR : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
-------------------------------------------------------------------
begin

  if CLAUSE_PTR = null then

    -- Start at the first clause in the knowledge-base

    CLAUSE_PTR := KB.FIRST;

   else

    -- Start at the last place a match was found for this goal

    CLAUSE_PTR := CLAUSE_PTR.NEXT_CLAUSE;

  end if;

  MATCHED := FALSE;

  while CLAUSE_PTR /= null
  loop

    if (CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE.ARITY =
       CLAUSE_PTR.ARITY)and then (SYSTEM_TYPES_PACKAGE.
                                  DYNAMIC_STRING.
                                  IS_EQUAL
                                  (CURRENT_GOAL_INSTANCE.
                                   GOAL.STRUCTURE.NAME,
                                   CLAUSE_PTR.NAME)) then

      -- Create a goal/match record pair

      CREATE(CURRENT_LEVEL, VARIABLES_PTR, CLAUSE_PTR);
      CURRENT_GOAL_INSTANCE.MATCH.INSTANCE := VARIABLES_PTR;
      CURRENT_GOAL_INSTANCE.MATCH.STRUCTURE := CLAUSE_PTR;
      MATCHED := TRUE;
      exit;

    end if;

    -- Try next clause

    CLAUSE_PTR := CLAUSE_PTR.NEXT_CLAUSE;

  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "MATCH");
-------------------------------------------------------------------
end MATCH;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- First sub-goal checks to see whether the current subgoal is the
-- first sub goal of a rule.   Used to direct backtracking when
-- removing subgoals from goals_to_solve.
--
-------------------------------------------------------------------
function FIRST_SUB_GOAL_OF_RULE
-------------------------------------------------------------------
(GOAL_PTR : in LOGIC_KB.KB_NODE_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin

  return (GOAL_PTR.KIND = LOGIC_KB_PACKAGE.FIRST_SUB_GOAL_NODE);

-------------------------------------------------------------------
end FIRST_SUB_GOAL_OF_RULE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Backtrack steps back to the previous goal.
-- If the current goal is the first subgoal of a rule then all
-- other subgoals of this rule are removed from goals to solve
-- before stepping back to the previous goal
--
-------------------------------------------------------------------
procedure BACKTRACK
-------------------------------------------------------------------
(CURRENT_INSTANCE : in out INSTANCE_RECORD_PTR;
 GOALS_TO_SOLVE   : in out GOALS.LIST_TYPE) is
-------------------------------------------------------------------
  GOAL_PTR : LOGIC_KB.KB_NODE_PTR_TYPE :=
             CURRENT_INSTANCE.GOAL.STRUCTURE;
-------------------------------------------------------------------
begin

  if FIRST_SUB_GOAL_OF_RULE(GOAL_PTR) then

    -- Delete all subgoals of this rule from goals_to_solve

    GOAL_PTR := GOAL_PTR.NEXT_GOAL;
    while GOAL_PTR /= null
    loop
      GOALS.DELETE_FRONT_OF(GOALS_TO_SOLVE);
      GOAL_PTR := GOAL_PTR.NEXT_GOAL;
    end loop;

    -- then exit which looses the first subgoal which is the
    -- current goal

  else

    -- delete match for this current goal
    -- and place it back on goals_to_solve

    CURRENT_INSTANCE.MATCH.INSTANCE := null;
    CURRENT_INSTANCE.MATCH.STRUCTURE := null;
    GOALS.PUT_ON_FRONT_OF(GOALS_TO_SOLVE, CURRENT_INSTANCE);
```

```
  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "BACKTRACK");
-------------------------------------------------------------------
end BACKTRACK;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Add_query_goals adds all the query gaols to gaols_to_solve
--
-------------------------------------------------------------------
procedure ADD_QUERY_GOALS
-------------------------------------------------------------------
(QUERY          : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 GOALS_TO_SOLVE : in out GOALS.LIST_TYPE) is
-------------------------------------------------------------------
  GOAL_PTR           : LOGIC_KB.KB_NODE_PTR_TYPE := QUERY;
  INSTANCE_VARIABLES_PTR : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
  GOAL_INSTANCE_PTR      : INSTANCE_RECORD_PTR;
-------------------------------------------------------------------
begin

  -- create an instance of the variables instance_template

  CREATE(0, INSTANCE_VARIABLES_PTR, QUERY );

  -- May be null if no variables
  -- create goal instances to put on goals_to_solve

  while GOAL_PTR /= null
  loop
    CREATE(GOAL_INSTANCE_PTR, GOAL_PTR, INSTANCE_VARIABLES_PTR);
    GOALS.PUT_ON_BACK_OF(GOALS_TO_SOLVE, GOAL_INSTANCE_PTR);
    GOAL_PTR := GOAL_PTR.NEXT_GOAL;
  end loop;

  -- Maintain a global pointer to the query variables so that they
  -- can be accessed from anywhere in the recursive stack when a
  -- solution is found

  QUERY_VARIABLES := INSTANCE_VARIABLES_PTR;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "ADD_QUERY_GOALS");
-------------------------------------------------------------------
end ADD_QUERY_GOALS;
-------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Add_sub_goals of a matched rule to goals_to_solve
--
------------------------------------------------------------------
procedure ADD_SUB_GOALS
------------------------------------------------------------------
(SUB_GOAL                : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 INSTANCE_VARIABLES_PTR : in     LOGIC_KB.
                                  INSTANCE_VARIABLES_RECORD_PTR;
 GOALS_TO_SOLVE          : in out GOALS.LIST_TYPE) is
------------------------------------------------------------------
  GOAL_PTR          : LOGIC_KB.KB_NODE_PTR_TYPE := SUB_GOAL;
  GOAL_INSTANCE_PTR : INSTANCE_RECORD_PTR;
  TEMP_QUEUE        : GOALS.LIST_TYPE;
------------------------------------------------------------------
begin

  -- Create tempoary queue of sub_goal instances

  while GOAL_PTR /= null
  loop
    CREATE(GOAL_INSTANCE_PTR, GOAL_PTR, INSTANCE_VARIABLES_PTR);
    GOALS.PUT_ON_FRONT_OF(TEMP_QUEUE, GOAL_INSTANCE_PTR);
    GOAL_PTR := GOAL_PTR.NEXT_GOAL;
  end loop;

  -- Add goal instances to front of goals_to_solve in the reverse
  -- order found in the knowledge-base e.g R :- S1, S2, S3
  -- added as Front -> S1 S2 S3 .... previous goals   <- Back

  while not GOALS.IS_EMPTY(TEMP_QUEUE)
  loop
    GOALS.GET_FROM_FRONT_OF(TEMP_QUEUE, GOAL_INSTANCE_PTR);
    GOALS.PUT_ON_FRONT_OF(GOALS_TO_SOLVE, GOAL_INSTANCE_PTR);
  end loop;
------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "ADD_SUB_GOALS");
------------------------------------------------------------------
end ADD_SUB_GOALS;
------------------------------------------------------------------


------------------------------------------------------------------
--
-- Or_reset is used to replace the current_goal_instance, which
-- should be the first subgoal of a rule, with the alternate 'or'
-- part of the rule.  The subgoals associated with the failed part
-- of the rule are removed from goals_to_solve and replaced by the
-- subgoals forming the 'or' part of the rule
--
------------------------------------------------------------------
procedure OR_RESET
------------------------------------------------------------------
(CURRENT_SUB_GOAL      : in     LOGIC_KB.KB_NODE_PTR_TYPE;
 CURRENT_GOAL_INSTANCE : in out INSTANCE_RECORD_PTR;
```

```
 GOALS_TO_SOLVE          : in out GOALS.LIST_TYPE;
 BIND_QUEUE               :in out BIND_PACKAGE.LIST_TYPE) is
-------------------------------------------------------------------
  -- Point at the current second subgaol since the first is on the
  -- recursive stack

  GOAL_PTR : LOGIC_KB.KB_NODE_PTR_TYPE :=
             CURRENT_SUB_GOAL.NEXT_GOAL;

  -- Point at the first 'or' subgoal

  FIRST_OR_GOAL : LOGIC_KB.KB_NODE_PTR_TYPE :=
                  CURRENT_SUB_GOAL.OR_SUB_GOAL;
-------------------------------------------------------------------
begin

  -- Delete current subgoals on goals to solve

  while GOAL_PTR /= null
  loop

    GOALS.DELETE_FRONT_OF(GOALS_TO_SOLVE);
    GOAL_PTR := GOAL_PTR.NEXT_GOAL;

  end loop;

  -- Unbind variables

  UNBIND(BIND_QUEUE);

  -- Reset goal instance to point to new first subgoal with no
  -- match

  CURRENT_GOAL_INSTANCE.MATCH.STRUCTURE := null;
  CURRENT_GOAL_INSTANCE.MATCH.INSTANCE := null;
  CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE := FIRST_OR_GOAL;

 -- Add all other 'or' subgoals to goals to solve

  if FIRST_OR_GOAL.NEXT_GOAL /= null then

    ADD_SUB_GOALS(FIRST_OR_GOAL.NEXT_GOAL,
                  CURRENT_GOAL_INSTANCE.GOAL.INSTANCE,
                  GOALS_TO_SOLVE);
  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "OR_RESET");
-------------------------------------------------------------------
end OR_RESET;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Is_built_in_operator checks current goal for built-in operation
--
```

```
-------------------------------------------------------------------
function IS_BUILT_IN_OPERATOR
-------------------------------------------------------------------
(CURRENT_GOAL_INSTANCE : in INSTANCE_RECORD_PTR)
return BOOLEAN is
-------------------------------------------------------------------
  OP: LOGIC_KB.KB_NODE_PTR_TYPE :=
      CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE;
-------------------------------------------------------------------
begin

  if SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.IS_EQUAL(OP.NAME, "\=") or else
     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.IS_EQUAL(OP.NAME, "is") or else
     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.IS_EQUAL(OP.NAME, "<") then -- or else other
                                                -- operations

    return TRUE;

  else

    return FALSE;

  end if;

-------------------------------------------------------------------
end IS_BUILT_IN_OPERATOR;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Built_in_operator Less_than
--
-------------------------------------------------------------------
FUNCTION LESS_THAN
(LEFT      : in LOGIC_KB.KB_NODE_PTR_TYPE;
 RIGHT     : in LOGIC_KB.KB_NODE_PTR_TYPE;
 VARIABLES : in LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR)
return BOOLEAN is
-------------------------------------------------------------------
  LEFT_BOUND_VALUE      : LOGIC_KB.KB_NODE_PTR_TYPE;
  LEFT_BOUND_VARIABLES  : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
  RIGHT_BOUND_VALUE     : LOGIC_KB.KB_NODE_PTR_TYPE;
  RIGHT_BOUND_VARIABLES : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR;
  LEFT_IS_VAR           : BOOLEAN;
  RIGHT_IS_VAR          : BOOLEAN;
  -----------------------------------------------------------------
  function "<"
  -----------------------------------------------------------------
  (LEFT  : in LOGIC_KB.KB_NODE_PTR_TYPE;
   RIGHT : in LOGIC_KB.KB_NODE_PTR_TYPE)
  return BOOLEAN is
  -----------------------------------------------------------------
  begin
    return STANDARD.INTEGER'
           VALUE
           (SYSTEM_TYPES_PACKAGE.
```

```
            DYNAMIC_STRING.
            SUBSTRING_OF
            (THE_STRING => LEFT.NAME))

           <

           STANDARD.INTEGER'
           VALUE
           (SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            SUBSTRING_OF
            (THE_STRING => RIGHT.NAME));
  ------------------------------------------------------------------
  end "<";
  ------------------------------------------------------------------
begin
  LEFT_IS_VAR  := IS_A_VARIABLE(LEFT);
  RIGHT_IS_VAR := IS_A_VARIABLE(RIGHT);
  if not LEFT_IS_VAR and then not RIGHT_IS_VAR then
    return LEFT < RIGHT;
  elsif LEFT_IS_VAR and then not RIGHT_IS_VAR then
    FIND
    (INSTANTIATED_STRUCTURE_PTR => LEFT_BOUND_VALUE,
     INSTANTIATED_VARIABLES_PTR => LEFT_BOUND_VARIABLES,
     STRUCTURE_PTR              => LEFT,
     VARIABLES_PTR              => VARIABLES);
    return LEFT_BOUND_VALUE < RIGHT;
  elsif not LEFT_IS_VAR and then RIGHT_IS_VAR then
    FIND
    (INSTANTIATED_STRUCTURE_PTR => RIGHT_BOUND_VALUE,
     INSTANTIATED_VARIABLES_PTR => RIGHT_BOUND_VARIABLES,
     STRUCTURE_PTR              => RIGHT,
     VARIABLES_PTR              => VARIABLES);
    return LEFT < RIGHT_BOUND_VALUE;
  else
    FIND
    (INSTANTIATED_STRUCTURE_PTR => LEFT_BOUND_VALUE,
     INSTANTIATED_VARIABLES_PTR => LEFT_BOUND_VARIABLES,
     STRUCTURE_PTR              => LEFT,
     VARIABLES_PTR              => VARIABLES);
    FIND
    (INSTANTIATED_STRUCTURE_PTR => RIGHT_BOUND_VALUE,
     INSTANTIATED_VARIABLES_PTR => RIGHT_BOUND_VARIABLES,
     STRUCTURE_PTR              => RIGHT,
     VARIABLES_PTR              => VARIABLES);
    return LEFT_BOUND_VALUE < RIGHT_BOUND_VALUE;

  end if;
------------------------------------------------------------------
end LESS_THAN;
------------------------------------------------------------------


------------------------------------------------------------------
--
-- Built_in_operator carries out the operation
--
------------------------------------------------------------------
procedure BUILT_IN_OPERATOR
------------------------------------------------------------------
```

```
(CURRENT_GOAL_INSTANCE : in      INSTANCE_RECORD_PTR;
 BIND_QUEUE            : in  out BIND_PACKAGE.LIST_TYPE;
 SATISFIED             : out     BOOLEAN) is
-------------------------------------------------------------------
  OP               : LOGIC_KB.KB_NODE_PTR_TYPE :=
                     CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE;
  LEFT_HAND_SIDE   : LOGIC_KB.
                     KB_NODE_PTR_TYPE := OP.NEXT_PARAMETER;
  RIGHT_HAND_SIDE  : LOGIC_KB.KB_NODE_PTR_TYPE :=
                     LEFT_HAND_SIDE.NEXT_GOAL;
  VARIABLES        : LOGIC_KB.INSTANCE_VARIABLES_RECORD_PTR :=
                     CURRENT_GOAL_INSTANCE.GOAL.INSTANCE;
  UNIFIED          : BOOLEAN := TRUE;
-------------------------------------------------------------------
begin

  if SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.IS_EQUAL(OP.NAME, "\=") then

    UNIFY_OPERANDS(LEFT_HAND_SIDE,
                   VARIABLES,
                   RIGHT_HAND_SIDE,
                   VARIABLES,
                   BIND_QUEUE,
                   UNIFIED);

    SATISFIED := not UNIFIED;

  elsif SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        IS_EQUAL
        (OP.NAME, "is") then

    UNIFY_OPERANDS(LEFT_HAND_SIDE,
                   VARIABLES,
                   RIGHT_HAND_SIDE,
                   VARIABLES,
                   BIND_QUEUE,
                   UNIFIED);

    SATISFIED := UNIFIED;

  elsif SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        IS_EQUAL
        (OP.NAME, "<") then

    SATISFIED := LESS_THAN
                 (LEFT      => LEFT_HAND_SIDE,
                  RIGHT     => RIGHT_HAND_SIDE,
                  VARIABLES => VARIABLES);

  -- elsif other operations

  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
```

```
     "BUILT_IN_OPERATOR");
-------------------------------------------------------------------
end BUILT_IN_OPERATOR;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Goal_is_cut checks the current goal to see whether it is the
-- cut
--
-------------------------------------------------------------------
function GOAL_IS_CUT
-------------------------------------------------------------------
(CURRENT_GOAL_INSTANCE : in     INSTANCE_RECORD_PTR)
 return BOOLEAN is
-------------------------------------------------------------------

  OP : LOGIC_KB.KB_NODE_PTR_TYPE :=
       CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE;
-------------------------------------------------------------------
begin

  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.IS_EQUAL(OP.NAME, "!");

-------------------------------------------------------------------
end GOAL_IS_CUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Goal_is_fail checks the current goal to see if it is fail
--
-------------------------------------------------------------------
function GOAL_IS_FAIL
-------------------------------------------------------------------
(CURRENT_GOAL_INSTANCE : in INSTANCE_RECORD_PTR) return BOOLEAN is
-------------------------------------------------------------------
  OP : LOGIC_KB.KB_NODE_PTR_TYPE :=
       CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE;
-------------------------------------------------------------------
begin

  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.IS_EQUAL(OP.NAME, "fail");

-------------------------------------------------------------------
end GOAL_IS_FAIL;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Resolve attempts to find a Match for the current goal.   If a
-- Match is found then an attempt is made to Unify the gaol and
-- match.   Should Unify fail then Match attempts to find other
-- matching structures lower down in the knowledge-base.   This
-- process continues until unify succeeds or no more matches can
```

```
-- be found.
--
-------------------------------------------------------------------
 procedure RESOLVE
 (KB                : in     LOGIC_KB.KB_RECORD;
  CURRENT_LEVEL     : in     NATURAL;
  CURRENT_INSTANCE  : in     INSTANCE_RECORD_PTR;
  GOALS_TO_SOLVE    : in out GOALS.LIST_TYPE;
  BIND_QUEUE        : in out BIND_PACKAGE.LIST_TYPE;
  IS_RESOLVED       : in out BOOLEAN) is
-------------------------------------------------------------------
  MATCHED : BOOLEAN;
  UNIFIED : BOOLEAN;
-------------------------------------------------------------------
begin

  loop

    MATCH( KB,
           CURRENT_LEVEL,
           CURRENT_INSTANCE,
           MATCHED);

    if MATCHED then

      UNIFIED := TRUE;
      UNIFY(CURRENT_INSTANCE.GOAL.STRUCTURE,
            CURRENT_INSTANCE.GOAL.INSTANCE,
            CURRENT_INSTANCE.MATCH.STRUCTURE,
            CURRENT_INSTANCE.MATCH.INSTANCE,
            --CURRENT_INSTANCE,
            BIND_QUEUE, UNIFIED);

      if UNIFIED then

        if IS_A_RULE(CURRENT_INSTANCE.MATCH.STRUCTURE) then

          -- Add all subgoals to goals_to_solve

          ADD_SUB_GOALS(CURRENT_INSTANCE.MATCH.
                        STRUCTURE.NEXT_GOAL,
                        CURRENT_INSTANCE.MATCH.INSTANCE,
                        GOALS_TO_SOLVE);
        end if;

        IS_RESOLVED := TRUE;
        exit;

      else

        -- Go around again and try further down the knowledge-base

        UNBIND(BIND_QUEUE);

      end if;

    else

      IS_RESOLVED := FALSE;
      exit;
```

```
    end if;

  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "RESOLVE");
-------------------------------------------------------------------
end RESOLVE;
-------------------------------------------------------------------

-------------------------------------------------------------------
--
-- Solve controls the inference process
-- A solution has been found when the current goal has been
-- satisfied and Goals_to_solve is empty
--
-------------------------------------------------------------------
task body SOLVE is
-------------------------------------------------------------------
  CURRENT_LEVEL    : NATURAL := 0;
  GOALS_TO_SOLVE   : GOALS.LIST_TYPE;
  CUT              : BOOLEAN := FALSE;
  CUT_LEVEL        : NATURAL := 0;
  SOLVED           : BOOLEAN := FALSE;
  FINISHED         : BOOLEAN := FALSE;

  -----------------------------------------------------------------
  --
  -- Solve_goals is a recursive subprogram which is called for
  -- each goal that has to be solved.
  -- On entry if goals_to_solve is empty then a solution has been
  -- found and result is reported otherwise the next goal is
  -- processed.
  --
  -----------------------------------------------------------------
  procedure SOLVE_GOALS
  (KB             : in     LOGIC_KB.KB_RECORD;
   CURRENT_LEVEL  : in     NATURAL;
   CUT            : in out BOOLEAN;
   CUT_LEVEL      : in out NATURAL;
   SOLVED         : in out BOOLEAN;
   FINISHED       : in out BOOLEAN) is
  -----------------------------------------------------------------
    BIND_QUEUE             : BIND_PACKAGE.LIST_TYPE;
    CURRENT_GOAL_INSTANCE  : INSTANCE_RECORD_PTR;
    MORE                   : BOOLEAN := FALSE;
    SATISFIED              : BOOLEAN := FALSE;
  -----------------------------------------------------------------
  begin

    if GOALS.IS_EMPTY(GOALS_TO_SOLVE) then

      -- A solution has been found

      declare
```

```
      VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING_LIST_PACKAGE.
                   LIST_TYPE;
    begin

      DISPLAY_SOLUTION(QUERY_VARIABLES);

    end;

      SOLVED := TRUE;
      FINISHED := FALSE;

  else

    -- Get next goal to satisfy

    GOALS.
    GET_FROM_FRONT_OF
    (GOALS_TO_SOLVE, CURRENT_GOAL_INSTANCE);
    loop

      if IS_BUILT_IN_OPERATOR(CURRENT_GOAL_INSTANCE) then

        -- This goal is a built-in operation

        BUILT_IN_OPERATOR(CURRENT_GOAL_INSTANCE,
                          BIND_QUEUE,
                          SATISFIED);

      elsif GOAL_IS_CUT(CURRENT_GOAL_INSTANCE) then

        -- Must be satisfied

        SATISFIED := TRUE;

      elsif GOAL_IS_FAIL(CURRENT_GOAL_INSTANCE) then

        -- fail

        BACKTRACK(CURRENT_GOAL_INSTANCE, GOALS_TO_SOLVE);
        CUT := FALSE;
        CUT_LEVEL := 0;
        FINISHED := FALSE;
        SOLVED := FALSE;
        exit;

      else

        -- Match and unify current goal

        RESOLVE(KB,
                CURRENT_LEVEL,
                CURRENT_GOAL_INSTANCE,
                GOALS_TO_SOLVE,
                BIND_QUEUE,
                SATISFIED);

      end if;
```

```
if SATISFIED then

  -- solve next goal

  SOLVE_GOALS(KB,
              CURRENT_LEVEL + 1,
              CUT,
              CUT_LEVEL,
              SOLVED, FINISHED);

  -- if cut not in progress

  if not CUT then

    -- but current goal is a cut

    if GOAL_IS_CUT(CURRENT_GOAL_INSTANCE) then

      CUT := TRUE;
      CUT_LEVEL := CURRENT_GOAL_INSTANCE.
                   GOAL.
                   INSTANCE.
                   LEVEL;

    end if;

  end if;


  if FINISHED then

    -- unwind recursion to quit

    exit;

  elsif SOLVED then

    -- A solution has just been reported. Any more?

    CONTROL.
    ANY_MORE(MORE);

     if MORE then

       UNBIND(BIND_QUEUE);
       FINISHED := FALSE;
       SOLVED := FALSE;

       if IS_BUILT_IN_OPERATOR(CURRENT_GOAL_INSTANCE) then

         -- Since the current_goal is a built-in operation
         -- search and match is not being used.
         -- Put this operation back on goals_to_solve and
         -- try
         -- previous goal again

         BACKTRACK(CURRENT_GOAL_INSTANCE, GOALS_TO_SOLVE);
         exit;
```

```
      elsif CUT then

        -- Current goal must be the cut
        -- eg A :- B, C, !.
        -- discard the gaol and start cutting

        exit;

      end if;

      -- Or go and try the current goal again further
      -- down the KB

    else

      -- No more solutions required then exit and quit

      FINISHED := TRUE;
      --SOLVED := FALSE; -- for interactive version which
                         -- produces which produces a
                         -- 'No'
                         -- on the screen
      SOLVED := TRUE;    -- for the embedded version
                         -- which inhibits the production
                         -- of'No' which would prevent
                         -- Solve from restarting.

      exit;

    end if;

  else

    -- Not finished and previous goal not solved. Try
    -- current goal further down the KB.

    UNBIND(BIND_QUEUE);
    FINISHED := FALSE;
    SOLVED := FALSE;

    if IS_BUILT_IN_OPERATOR(CURRENT_GOAL_INSTANCE) then

      -- Since the current_goal is a built-in operation
      -- search and match is not being used.
      -- Put this operation back on goals_to_solve and try
      -- previous goal again

      BACKTRACK(CURRENT_GOAL_INSTANCE, GOALS_TO_SOLVE);
      exit;

    elsif CUT then

      if CURRENT_LEVEL /= CUT_LEVEL then

        exit;

      else

        -- Parent clause found
```

```
            BACKTRACK(CURRENT_GOAL_INSTANCE, GOALS_TO_SOLVE);
            CUT := FALSE;
            CUT_LEVEL := 0;

            exit;

          end if;

        end if;

      end if;

    else

      -- Current goal not satisified by complete search of KB.

      if FIRST_SUB_GOAL_OF_RULE
         (CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE) then
        if CURRENT_GOAL_INSTANCE.
           GOAL.
           STRUCTURE.
           OR_SUB_GOAL /=  null then

          -- Current goal not satisified by complete search of
          -- KB.
          -- Current goal is the first sub goal of a rule
          -- and there is an 'or' set of subgoals

          OR_RESET(CURRENT_GOAL_INSTANCE.GOAL.STRUCTURE,
                   CURRENT_GOAL_INSTANCE,
                   GOALS_TO_SOLVE,
                   BIND_QUEUE);

          FINISHED := FALSE;
          SOLVED := FALSE;

          -- try resolve again with new 'or' subgoals at
          -- current level

        else

          -- Current goal not satisified by complete search of
          -- KB.
          -- Current goal is the first sub goal of a rule
          -- but there is no 'or' set of subgoals
          -- Backtrack from this point by pushing this goal
          -- back onto goals_to_solve then drop down to
          -- previous goal to continue the search

          UNBIND(BIND_QUEUE);
          BACKTRACK(CURRENT_GOAL_INSTANCE, GOALS_TO_SOLVE);
          FINISHED := FALSE;
          SOLVED := FALSE;
          exit;

        end if;

      else

        -- Current goal not satisified by complete search of
```

```
            -- KB.
            -- It is not the first subgoal of a rule
            -- Backtrack from this point by pushing this goal back
            -- onto goals_to_solve then drop down to previous goal
            -- to continue the search

            UNBIND(BIND_QUEUE);
            BACKTRACK(CURRENT_GOAL_INSTANCE, GOALS_TO_SOLVE);
            FINISHED := FALSE;
            SOLVED := FALSE;
            exit;

          end if;

        end if;

       end loop;

    end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "SOLVE_GOALS");
  -----------------------------------------------------------------
  end SOLVE_GOALS;
  -----------------------------------------------------------------

-------------------------------------------------------------------
begin

  loop

    declare

      QUERY : LOGIC_KB.KB_RECORD;
      KB    : LOGIC_KB.KB_RECORD;

    begin
      select
        accept START(THIS_QUERY : in LOGIC_KB.KB_RECORD;
                     THIS_KB    : in LOGIC_KB.KB_RECORD) do

          QUERY := THIS_QUERY;
          KB    := THIS_KB;

        end START;
      or
        terminate;
      end select;

    -- Reset

    CURRENT_LEVEL := 0;
    GOALS.CLEAR(LIST => GOALS_TO_SOLVE);
    CUT := FALSE;
    CUT_LEVEL := 0;
    SOLVED := FALSE;
    FINISHED := FALSE;
```

```
      ADD_QUERY_GOALS(QUERY.FIRST, GOALS_TO_SOLVE);

      SOLVE_GOALS(KB,
                  CURRENT_LEVEL + 1,
                  CUT,
                  CUT_LEVEL,
                  SOLVED,
                  FINISHED);

      if not SOLVED then

        declare
          NO : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               STRING;

          VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING_LIST_PACKAGE.
                       LIST_TYPE;

        begin

          SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
          COPY
          (FROM_THE_SUBSTRING => "No",
           TO_THE_STRING => NO);

          SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING_LIST_PACKAGE.
          PUT_ON_BACK_OF
          (LIST => VALUE_LIST,
           THIS_ITEM => NO);

          CONTROL.
          PUT_RESULT
          (IN_LIST => VALUE_LIST);

        end;

      end if;

    end;

  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
     "SOLVE");
-------------------------------------------------------------------
end SOLVE;
-------------------------------------------------------------------


-------------------------------------------------------------------
task body CONTROL is
-------------------------------------------------------------------
```

```
begin
  loop
    select
      accept PUT_RESULT(IN_LIST : in SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING_LIST_PACKAGE.
                                    LIST_TYPE) do

        accept GET_RESULT
              (OUT_LIST : out SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING_LIST_PACKAGE.
                              LIST_TYPE) do

          OUT_LIST := IN_LIST;

        end GET_RESULT;

      end PUT_RESULT;

    or

      accept ANY_MORE(MORE : out BOOLEAN) do

        select
          accept GET_MORE do
           MORE := TRUE;
          end GET_MORE;
        or
          accept NO_MORE do
            MORE := FALSE;
          end NO_MORE;
        or terminate;
        end select;
      end ANY_MORE;
  or terminate;
  end select;

  end loop;
-------------------------------------------------------------------
  exception
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS raised in LOGIC_INFERENCE_PACKAGE" &
       "at CONTROL.");
-------------------------------------------------------------------
end CONTROL;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Test the inference strategy
--
-------------------------------------------------------------------
procedure TEST is
-------------------------------------------------------------------
  package INTEGER_TEXT_IO is new TEXT_IO.INTEGER_IO(INTEGER);
  use INTEGER_TEXT_IO;

  KB              : LOGIC_KB.KB_RECORD;
  QUERY           : LOGIC_KB.KB_RECORD;
```

```
   IN_QUERY         : STANDARD.STRING(1 .. 80);
   LENGTH_IN_QUERY  : NATURAL;
   REPLY            : STRING(1..3);
   LENGTH_REPLY     : NATURAL;
-------------------------------------------------------------------
begin

   TEXT_IO.PUT_LINE("Building Knowledge-Base");
   LOGIC_KB.BUILD(KB, "testkb.txt");
   TEXT_IO.PUT_LINE("Finished building Knowledge-Base");

   loop

     TEXT_IO.PUT_LINE("?-");

     TEXT_IO.GET_LINE(IN_QUERY, LENGTH_IN_QUERY);

     exit when IN_QUERY(1 .. LENGTH_IN_QUERY) = "stop.";

     TEXT_IO.PUT_LINE("Building the Query");
     LOGIC_KB.ASSERT(IN_CLAUSE  => IN_QUERY(1  .. LENGTH_IN_QUERY),
                     KB         => QUERY);

     TEXT_IO.PUT_LINE("Finished building Query");

     TEXT_IO.PUT_LINE("Starting to solve");

     SOLVE.START(THIS_QUERY => QUERY,
                 THIS_KB    => KB);

     TEXT_IO.PUT_LINE("Finished solving");

     OUTER:
     loop

       declare

         VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING_LIST_PACKAGE.
                      LIST_TYPE;

         ITEM : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;

       begin

         TEXT_IO.PUT_LINE("Getting the result");

         CONTROL.GET_RESULT(OUT_LIST => VALUE_LIST);

         TEXT_IO.PUT_LINE("Result returned");

         for I in 1 .. SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING_LIST_PACKAGE.
                       LENGTH_OF
                       (LIST => VALUE_LIST)
         loop

           SYSTEM_TYPES_PACKAGE.
```

```
            DYNAMIC_STRING_LIST_PACKAGE.
            GET_FROM_FRONT_OF
            (LIST => VALUE_LIST,
             THIS_ITEM => ITEM);

            SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            PUT
            (THE_STRING => ITEM);

            exit OUTER when SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            IS_EQUAL
                            (LEFT  => ITEM,
                             RIGHT => "No");

            TEXT_IO.PUT(" ");

          end loop;

        end;

        TEXT_IO.NEW_LINE;
        TEXT_IO.PUT_LINE("Finished printing the result");

        TEXT_IO.NEW_LINE;
        TEXT_IO.PUT_LINE("More? y/n : ");
        TEXT_IO.GET_LINE(REPLY, LENGTH_REPLY);

        if REPLY(1) = 'Y' or REPLY(1) = 'y' then
           TEXT_IO.PUT_LINE("Getting more");
           CONTROL.
           GET_MORE;
        else
           TEXT_IO.PUT_LINE("No more wanted");
           CONTROL.
           NO_MORE;
        end if;

      end loop OUTER;

      TEXT_IO.NEW_LINE;
      TEXT_IO.PUT_LINE("Retracting query");

      LOGIC_KB.
      RETRACT(CLAUSE => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => QUERY.
                                       FIRST.
                                       CLAUSE),
              KB     => QUERY);

      TEXT_IO.PUT_LINE("Query retracted");

    end loop;

    TEXT_IO.PUT_LINE("Test complete");
-------------------------------------------------------------------
exception
```

```
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in LOGIC_INFERENCE_PACKAGE at " &
       "TEST");
  ------------------------------------------------------------------
  end TEST;
  ------------------------------------------------------------------

------------------------------------------------------------------
begin

  null;

------------------------------------------------------------------
end LOGIC_INFERENCE_PACKAGE;
------------------------------------------------------------------
```

Rule Base Types

```
------------------------------------------------------------------
--
-- Unit        : RULE_BASE_TYPES_PACKAGE specification
-- Author      : A Harrison Software Engineering Group,
--               Cranfield University, RMCS, Shrivenam
-- Date        : 10 January 1992
-- Function    : This package provides the types to build a rule-base
--
------------------------------------------------------------------
with SYSTEM_TYPES_PACKAGE,
   GENERIC_LIST_PACKAGE;
------------------------------------------------------------------
package RULE_BASE_TYPES_PACKAGE is
------------------------------------------------------------------

  type NODE_TYPE is (HEAD_NODE, ANTECEDENT, CONSEQUENT);

  type RULE_BASE_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE);

  type RULE_BASE_NODE_PTR_TYPE is access RULE_BASE_NODE_RECORD;

  type RULE_BASE_NODE_RECORD(KIND : NODE_TYPE := HEAD_NODE) is
  record

    case KIND is
      when HEAD_NODE =>

        RULE_NUMBER : POSITIVE;
        FIRED       : BOOLEAN;
        CONDITIONS  : RULE_BASE_NODE_PTR_TYPE;
        ACTION      : RULE_BASE_NODE_PTR_TYPE;
        NEXT_RULE   : RULE_BASE_NODE_PTR_TYPE;

      when ANTECEDENT =>

        FACT      : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        NEXT_FACT : RULE_BASE_NODE_PTR_TYPE;

      when CONSEQUENT =>

        ACTION_TO_DO : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;

    end case;
  end record;

  ------------------------------------------------------------------

  type RULE_BASE_RECORD;
  type RULE_BASE_RECORD_PTR_TYPE is access RULE_BASE_RECORD;
  type RULE_BASE_RECORD is
  record
    RULE_BASE_NAME : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
    FIRST_RULE     : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;
    NEXT_RULE_BASE : RULE_BASE_RECORD_PTR_TYPE;
  end record;
```

```
  ------------------------------------------------------------------------

   function IS_EQUAL
   (RULE_BASE_NODE_PTR_1 : in RULE_BASE_NODE_PTR_TYPE;
    RULE_BASE_NODE_PTR_2 : in RULE_BASE_NODE_PTR_TYPE)
   return BOOLEAN;

   package AGENDA_LIST_PACKAGE is new GENERIC_LIST_PACKAGE
   (RULE_BASE_NODE_PTR_TYPE,
    IS_EQUAL);

------------------------------------------------------------------------
end RULE_BASE_TYPES_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit        : RULE_BASE_TYPES_PACKAGE body
-- Author      : A Harrison Software Engineering Group,
--               Cranfield University, RMCS, Shrivenam
-- Date        : 10 January 1992
-- Function    : This package provides the types to build a rule-base
--
-------------------------------------------------------------------
package body RULE_BASE_TYPES_PACKAGE is
-------------------------------------------------------------------

  -----------------------------------------------------------------
  function IS_EQUAL
  -----------------------------------------------------------------
  (RULE_BASE_NODE_PTR_1 : in RULE_BASE_NODE_PTR_TYPE;
   RULE_BASE_NODE_PTR_2 : in RULE_BASE_NODE_PTR_TYPE)
  return BOOLEAN is
  -----------------------------------------------------------------
  begin

    return RULE_BASE_NODE_PTR_1.RULE_NUMBER = RULE_BASE_NODE_PTR_2.
                                              RULE_NUMBER;

  -----------------------------------------------------------------
  end IS_EQUAL;
  -----------------------------------------------------------------


-------------------------------------------------------------------
end RULE_BASE_TYPES_PACKAGE;
-------------------------------------------------------------------
```

Rule Base

```
------------------------------------------------------------------------
--
-- Unit      : GENERIC_RULE_BASE_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 14 December 1991
-- Function  : This package provides the operations to build a
--             rule-base
--
------------------------------------------------------------------------
with TEXT_IO,
     RULE_BASE_TYPES_PACKAGE,
     SYSTEM_TYPES_PACKAGE;
------------------------------------------------------------------------
generic
  RULE_BASES_FILENAME : in STANDARD.STRING;
------------------------------------------------------------------------
package GENERIC_RULE_BASE_PACKAGE is
------------------------------------------------------------------------
  RULE_BASES_PTR : RULE_BASE_TYPES_PACKAGE.
  RULE_BASE_RECORD_PTR_TYPE;
  ----------------------------------------------------------------------

  procedure BUILD
  (RULE_BASES_PTR           : in out RULE_BASE_TYPES_PACKAGE.
                                     RULE_BASE_RECORD_PTR_TYPE;
   RULE_BASES_FILENAME      : in     STANDARD.STRING);

  procedure FIND
  (RULE_BASE : in     STANDARD.STRING;
   RULE_PTR  :    out RULE_BASE_TYPES_PACKAGE.
                      RULE_BASE_NODE_PTR_TYPE);

  procedure PRINT
  (RULE : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE);

  procedure PRINT
  (RULE_BASES_PTR : in RULE_BASE_TYPES_PACKAGE.
                       RULE_BASE_RECORD_PTR_TYPE);

  function IS_EQUAL
  (NODE_PTR_1 : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;
   NODE_PTR_2 : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE)
  return BOOLEAN;

  procedure TEST;

------------------------------------------------------------------------
end GENERIC_RULE_BASE_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : GENERIC_RULE_BASE_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 14 December 1991
-- Function  : This package provides the operations to build a
--             rule-base
--
-------------------------------------------------------------------
package body GENERIC_RULE_BASE_PACKAGE is
-------------------------------------------------------------------
use RULE_BASE_TYPES_PACKAGE; -- Otherwise it can't sort out the
                             -- operations = and /= etc
  -----------------------------------------------------------------
  --
  -- Construct collects all the ANTECEDENTs and links them to a new
  -- rule_head.  The consequence is then collected and also linked
  -- to the rule_head.  The complete rule is returned
  --
  -----------------------------------------------------------------
  procedure CONSTRUCT
  -----------------------------------------------------------------
  (RULE     :    out RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;
   FILENAME : in     TEXT_IO.FILE_TYPE) is
  -----------------------------------------------------------------
    TOKEN          : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
    ANTECEDENT_PTR : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;
    CONSEQUENT_PTR : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;
    TEMP_PTR       : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;
    TEMP_RULE      : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;
  -----------------------------------------------------------------
    SYNTAX_ERROR : exception;
  -----------------------------------------------------------------
  begin

    -- get new rule head

    TEMP_RULE := new RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_RECORD;
    TEMP_RULE.FIRED := FALSE;

    SYSTEM_TYPES_PACKAGE.GET(TOKEN, FILENAME);
    if SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.IS_EQUAL(TOKEN, "IF") then

      -- Form first ANTECEDENT and link to the rule head

      SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.CLEAR(TOKEN);
      SYSTEM_TYPES_PACKAGE.GET(TOKEN, FILENAME);
      ANTECEDENT_PTR := new RULE_BASE_TYPES_PACKAGE.
                            RULE_BASE_NODE_RECORD
                            (RULE_BASE_TYPES_PACKAGE.ANTECEDENT);
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.COPY(TOKEN, ANTECEDENT_PTR.FACT);
      TEMP_RULE.CONDITIONS := ANTECEDENT_PTR;
```

```
TEMP_PTR := ANTECEDENT_PTR;

-- Iterate over remaining ANTECEDENTs connecting each to the
-- previous ANTECEDENT

loop

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  CLEAR(TOKEN);

  SYSTEM_TYPES_PACKAGE.
  GET(TOKEN, FILENAME);

  exit when SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            IS_EQUAL(TOKEN, "THEN");

  if not SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL(TOKEN, "AND") then

    TEXT_IO.PUT_LINE(RULE_BASES_FILENAME);
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    PUT(TOKEN);
    TEXT_IO.NEW_LINE;
    raise SYNTAX_ERROR;

  end if;

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  CLEAR(TOKEN);

  SYSTEM_TYPES_PACKAGE.
  GET(TOKEN, FILENAME);

  ANTECEDENT_PTR := new RULE_BASE_TYPES_PACKAGE.
                        RULE_BASE_NODE_RECORD
                        (RULE_BASE_TYPES_PACKAGE.ANTECEDENT);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY(TOKEN, ANTECEDENT_PTR.FACT);

  TEMP_PTR.NEXT_FACT := ANTECEDENT_PTR;
  TEMP_PTR := TEMP_PTR.NEXT_FACT;

end loop;

-- Add the consequent

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
CLEAR(TOKEN);

SYSTEM_TYPES_PACKAGE.
GET(TOKEN, FILENAME);
```

```
      CONSEQUENT_PTR := new RULE_BASE_TYPES_PACKAGE.
                            RULE_BASE_NODE_RECORD
                            (RULE_BASE_TYPES_PACKAGE.CONSEQUENT);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY(TOKEN, CONSEQUENT_PTR.ACTION_TO_DO);

      TEMP_RULE.ACTION := CONSEQUENT_PTR;

    else

      TEXT_IO.PUT_LINE(RULE_BASES_FILENAME);
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      PUT(TOKEN);
      TEXT_IO.NEW_LINE;
      raise SYNTAX_ERROR;

    end if;

    RULE := TEMP_RULE;
-------------------------------------------------------------------
exception
    when STORAGE_ERROR =>
      TEXT_IO.
      PUT_LINE("Heap overflow in RULE_BASE_PACKAGE.CONSTRUCT");
    when SYNTAX_ERROR =>
      TEXT_IO.
      PUT_LINE("Syntax error in RULE_BASE_PACKAGE.CONSTRUCT");
-------------------------------------------------------------------
end CONSTRUCT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Build the rule_base.  Return the completed rule_base
--
-------------------------------------------------------------------
procedure BUILD
-------------------------------------------------------------------
(RULE_BASES_PTR      : in out RULE_BASE_TYPES_PACKAGE.
                             RULE_BASE_RECORD_PTR_TYPE;
 RULE_BASES_FILENAME : in     STANDARD.STRING) is
  -----------------------------------------------------------------
  RULE_BASES_FILE             : TEXT_IO.FILE_TYPE;
  CURRENT_RULE_BASE_FILE      : TEXT_IO.FILE_TYPE;
  CURRENT_RULE_BASE_FILENAME  : SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                STRING;
  NEW_RULE_BASE_PTR           : RULE_BASE_TYPES_PACKAGE.
                                RULE_BASE_RECORD_PTR_TYPE;
  CURRENT_RULE_BASE_PTR       : RULE_BASE_TYPES_PACKAGE.
                                RULE_BASE_RECORD_PTR_TYPE;
  NEW_RULE_PTR                : RULE_BASE_TYPES_PACKAGE.
                                RULE_BASE_NODE_PTR_TYPE;
  CURRENT_RULE_PTR            : RULE_BASE_TYPES_PACKAGE.
                                RULE_BASE_NODE_PTR_TYPE;
-------------------------------------------------------------------
```

```
begin

  TEXT_IO.OPEN(FILE => RULE_BASES_FILE,
               MODE => TEXT_IO.IN_FILE,
               NAME => RULE_BASES_FILENAME);

  while not TEXT_IO.END_OF_FILE(RULE_BASES_FILE)
  loop
    if not SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_NULL(CURRENT_RULE_BASE_FILENAME) then

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      CLEAR(CURRENT_RULE_BASE_FILENAME);

    end if;

    SYSTEM_TYPES_PACKAGE.
    GET(TOKEN     => CURRENT_RULE_BASE_FILENAME,
        FROM_FILE => RULE_BASES_FILE);

    NEW_RULE_BASE_PTR := new RULE_BASE_TYPES_PACKAGE.
                             RULE_BASE_RECORD;

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    PREPEND
    (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                      GET_FIELD
                      (NUMBER => 1,
                       FROM   => SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING.
                                 SUBSTRING_OF
                                 (THE_STRING =>
                                  CURRENT_RULE_BASE_FILENAME)),
     TO_THE_STRING => NEW_RULE_BASE_PTR.RULE_BASE_NAME);

    if RULE_BASES_PTR = null then
      RULE_BASES_PTR := NEW_RULE_BASE_PTR;
      CURRENT_RULE_BASE_PTR := NEW_RULE_BASE_PTR;
    else
      CURRENT_RULE_BASE_PTR.NEXT_RULE_BASE := NEW_RULE_BASE_PTR;
      CURRENT_RULE_BASE_PTR := NEW_RULE_BASE_PTR;
    end if;

    TEXT_IO.
    OPEN(FILE => CURRENT_RULE_BASE_FILE,
         MODE => TEXT_IO.IN_FILE,
         NAME => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => CURRENT_RULE_BASE_FILENAME));

    while not TEXT_IO.END_OF_FILE(CURRENT_RULE_BASE_FILE)
    loop

      CONSTRUCT(RULE     => NEW_RULE_PTR,
                FILENAME => CURRENT_RULE_BASE_FILE);
```

```
      if CURRENT_RULE_BASE_PTR.FIRST_RULE = null then
        CURRENT_RULE_BASE_PTR.FIRST_RULE := NEW_RULE_PTR;
        CURRENT_RULE_PTR := NEW_RULE_PTR;
      else
        CURRENT_RULE_PTR.NEXT_RULE := NEW_RULE_PTR;
        CURRENT_RULE_PTR := NEW_RULE_PTR;
      end if;

    end loop;

    TEXT_IO.CLOSE(CURRENT_RULE_BASE_FILE);

  end loop;

  TEXT_IO.CLOSE(RULE_BASES_FILE);

-------------------------------------------------------------------
end BUILD;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Find the appropriate rule-base from the partitioned rule-bases
--
-------------------------------------------------------------------
procedure FIND
-------------------------------------------------------------------
(RULE_BASE : in      STANDARD.STRING;
 RULE_PTR  :    out RULE_BASE_TYPES_PACKAGE.
                    RULE_BASE_NODE_PTR_TYPE) is
-------------------------------------------------------------------

  TEMP_RULE_PTR : RULE_BASE_TYPES_PACKAGE.
                  RULE_BASE_RECORD_PTR_TYPE := RULE_BASES_PTR;

  RULE_BASE_NOT_FOUND : exception;

-------------------------------------------------------------------
begin

  while TEMP_RULE_PTR /= null
  loop

    if RULE_BASE(1) =
       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       ITEM_OF
       (THE_STRING      => TEMP_RULE_PTR.RULE_BASE_NAME,
        AT_THE_POSITION => 1) then

      RULE_PTR := TEMP_RULE_PTR.FIRST_RULE;
      return;

    else

      TEMP_RULE_PTR := TEMP_RULE_PTR.NEXT_RULE_BASE;

    end if;
```

```
  end loop;

  raise RULE_BASE_NOT_FOUND;
-------------------------------------------------------------------
exception
  when RULE_BASE_NOT_FOUND =>
    TEXT_IO.PUT_LINE
    ("Exception RULE_BASE_NOT_FOUND raised at FIND in " &
     "at GENERIC_RULE_BASE_PACKAGE");
-------------------------------------------------------------------
end FIND;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Print a single rule
--
-------------------------------------------------------------------
procedure PRINT
-------------------------------------------------------------------
  (RULE : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  FACT_PTR : RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE :=
             RULE.CONDITIONS;
-------------------------------------------------------------------
begin

  TEXT_IO.PUT("IF");
  TEXT_IO.SET_COL(6);
  while FACT_PTR /= null
  loop

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    PUT(FACT_PTR.FACT);

    if  FACT_PTR.NEXT_FACT /= null then
      TEXT_IO.PUT(" AND ");
    end if;

    FACT_PTR := FACT_PTR.NEXT_FACT;

  end loop;

  TEXT_IO.NEW_LINE;
  TEXT_IO.PUT("THEN ");
  TEXT_IO.SET_COL(6);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  PUT(RULE.ACTION.ACTION_TO_DO);

  TEXT_IO.NEW_LINE(2);

-------------------------------------------------------------------
end PRINT;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Print a the rule_base
--
-------------------------------------------------------------------
procedure PRINT
-------------------------------------------------------------------
(RULE_BASES_PTR : in RULE_BASE_TYPES_PACKAGE.
                    RULE_BASE_RECORD_PTR_TYPE) is
-------------------------------------------------------------------

  RULE_BASE_PTR : RULE_BASE_TYPES_PACKAGE.
                  RULE_BASE_RECORD_PTR_TYPE := RULE_BASES_PTR;

  RULE_PTR      : RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;
-------------------------------------------------------------------
begin

  while RULE_BASE_PTR /= null
  loop

    RULE_PTR := RULE_BASE_PTR.FIRST_RULE;
    while RULE_PTR /= null
    loop

      PRINT(RULE_PTR);
      RULE_PTR := RULE_PTR.NEXT_RULE;

    end loop;

    RULE_BASE_PTR := RULE_BASE_PTR.NEXT_RULE_BASE;

  end loop;

-------------------------------------------------------------------
end PRINT;
-------------------------------------------------------------------



-------------------------------------------------------------------
--
-- Is_equal tests content of nodes for equality. Used for
-- instantiation of
-- Agenda, but not used!
--
-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(NODE_PTR_1 : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;
 NODE_PTR_2 : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin

  return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.IS_EQUAL
         (NODE_PTR_1.FACT, NODE_PTR_2.FACT);

-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------
```

```
  ------------------------------------------------------------------
  --
  -- Test the building process
  --
  ------------------------------------------------------------------
  procedure TEST is
  ------------------------------------------------------------------
  begin

    PRINT(RULE_BASES_PTR);

  ------------------------------------------------------------------
  end TEST;
  ------------------------------------------------------------------

------------------------------------------------------------------
begin

  BUILD(RULE_BASES_PTR, RULE_BASES_FILENAME);

------------------------------------------------------------------
end GENERIC_RULE_BASE_PACKAGE;
------------------------------------------------------------------
```

Fact Base

```
------------------------------------------------------------------
--
-- Unit      : GENERIC_FACT_BASE_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 14 December 1991
-- Function  : This package provides the operations to build a
--             fact-base
--
------------------------------------------------------------------
with GENERIC_TREE_PACKAGE,
     SYSTEM_TYPES_PACKAGE,
     TEXT_IO;
------------------------------------------------------------------
generic

  type FACT_COMPOSITE_TYPE is private;
  type FACT_PTR_TYPE is access FACT_COMPOSITE_TYPE;

  with function "<"
       (LEFT  : in FACT_PTR_TYPE;
        RIGHT : in FACT_PTR_TYPE) return BOOLEAN;

  with function ">"
       (LEFT  : in FACT_PTR_TYPE;
        RIGHT : in FACT_PTR_TYPE) return BOOLEAN;

  with function IS_EQUAL
       (FACT     : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FACT_PTR : in FACT_PTR_TYPE) return BOOLEAN;

  with function IS_LESS_THAN
       (FACT     : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FACT_PTR : in FACT_PTR_TYPE) return BOOLEAN;

  with function IS_GREATER_THAN
       (FACT     : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FACT_PTR : in FACT_PTR_TYPE) return BOOLEAN;

  with procedure PUT
       (FACT : in FACT_PTR_TYPE);

------------------------------------------------------------------
package GENERIC_FACT_BASE_PACKAGE is
------------------------------------------------------------------

  package FACT_TREE_PACKAGE is new GENERIC_TREE_PACKAGE
    (ITEM_BASE_TYPE      => SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.STRING,
     ITEM_COMPOSITE_TYPE => FACT_COMPOSITE_TYPE,
     ITEM_PTR_TYPE       => FACT_PTR_TYPE,
     "<"                 => "<",
     ">"                 => ">",
     IS_EQUAL            => IS_EQUAL,
     IS_LESS_THAN        => IS_LESS_THAN,
     IS_GREATER_THAN     => IS_GREATER_THAN,
     PUT                 => PUT);
```

```
  type FACT_BASE_RECORD is
  record
    FACTS : FACT_TREE_PACKAGE.TREE_PTR_TYPE;
  end record;

  ------------------------------------------------------------------------
  FACT_BASE : FACT_BASE_RECORD;
  ------------------------------------------------------------------------

  procedure INSERT
  (FACT      : in FACT_PTR_TYPE;
   FACT_BASE : in out FACT_BASE_RECORD);

  function IS_IN
  (FACT      : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACT_BASE : in FACT_BASE_RECORD) return BOOLEAN;

  procedure PRINT
  (FACT_BASE : in FACT_BASE_RECORD);

------------------------------------------------------------------------
end GENERIC_FACT_BASE_PACKAGE;
------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unit      : GENERIC_FACT_BASE_PACKAGE body
-- Author    : A Harrison SofTware Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 14 December 1991
-- Function : This package provides the operations to build a
--             fact-base
--
------------------------------------------------------------------
package body GENERIC_FACT_BASE_PACKAGE is
  ----------------------------------------------------------------
  --
  -- Is_in determines whether a fact is in the fact base
  --
  ----------------------------------------------------------------
  function IS_IN
  ----------------------------------------------------------------
  (FACT      : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACT_BASE : in FACT_BASE_RECORD) return BOOLEAN is
  ----------------------------------------------------------------
  begin

  . return FACT_TREE_PACKAGE.IS_IN(FACT, FACT_BASE.FACTS);

  ----------------------------------------------------------------
  end IS_IN;
  ----------------------------------------------------------------


  ----------------------------------------------------------------
  --
  -- Insert a fact into the fact base
  --
  ----------------------------------------------------------------
  procedure INSERT
  ----------------------------------------------------------------
  (FACT      : in FACT_PTR_TYPE;
   FACT_BASE : in out FACT_BASE_RECORD) is
  ----------------------------------------------------------------
  begin

    FACT_TREE_PACKAGE.INSERT(FACT, FACT_BASE.FACTS);

  ----------------------------------------------------------------
  end INSERT;
  ----------------------------------------------------------------


  ----------------------------------------------------------------
  --
  -- Print the fact base
  --
  ----------------------------------------------------------------
  procedure PRINT
  ----------------------------------------------------------------
  (FACT_BASE : in FACT_BASE_RECORD) is
  ----------------------------------------------------------------
  begin
```

```
    FACT_TREE_PACKAGE.PRINT(FACT_BASE.FACTS);
  ------------------------------------------------------------------------
  end PRINT;
  ------------------------------------------------------------------------

------------------------------------------------------------------------
end GENERIC_FACT_BASE_PACKAGE;
------------------------------------------------------------------------
```

Rule Inference Engine

```
------------------------------------------------------------------
--
-- Unit      : GENERIC_RULE_BASE_INFERENCE_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 14 December 1991
-- Function  : This package provides the operations to forward chain
--             over a rule-base
------------------------------------------------------------------
with SYSTEM_TYPES_PACKAGE,
     RULE_BASE_TYPES_PACKAGE,
     GENERIC_RULE_BASE_PACKAGE,
     GENERIC_FACT_BASE_PACKAGE;
------------------------------------------------------------------
generic

  RULE_BASES_FILENAME : STANDARD.STRING;

  type FACT_COMPOSITE_TYPE is private;
  type FACT_PTR_TYPE is access FACT_COMPOSITE_TYPE;

  with function "<"
       (LEFT  : in FACT_PTR_TYPE;
        RIGHT : in FACT_PTR_TYPE) return BOOLEAN;

  with function ">"
       (LEFT  : in FACT_PTR_TYPE;
        RIGHT : in FACT_PTR_TYPE) return BOOLEAN;

   with function IS_EQUAL
        (FACT     : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
         FACT_PTR : in FACT_PTR_TYPE) return BOOLEAN;

  with function IS_LESS_THAN
       (FACT     : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FACT_PTR : in FACT_PTR_TYPE) return BOOLEAN;

  with function IS_GREATER_THAN
       (FACT     : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
        FACT_PTR : in FACT_PTR_TYPE) return BOOLEAN;

  with procedure PUT
       (FACT : in FACT_PTR_TYPE);

------------------------------------------------------------------
package GENERIC_RULE_BASE_INFERENCE_PACKAGE is
------------------------------------------------------------------

  procedure INFERENCE
  (RULE_BASE : in     STANDARD.STRING;
   FACT_PTR  : in     FACT_PTR_TYPE;
   AGENDA    : in out RULE_BASE_TYPES_PACKAGE.
                      AGENDA_LIST_PACKAGE.LIST_TYPE);

------------------------------------------------------------------
end GENERIC_RULE_BASE_INFERENCE_PACKAGE;
------------------------------------------------------------------
```

```
------------------------------------------------------------------------
--
-- Unit      : GENERIC_RULE_BASE_INFERENCE_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 14 December 1991
-- Function  : This package provides the operations to forward chain
--             over a rule-base
--
------------------------------------------------------------------------
with TEXT_IO;
------------------------------------------------------------------------
package body GENERIC_RULE_BASE_INFERENCE_PACKAGE is
------------------------------------------------------------------------

  package RULE_BASE_PACKAGE is new GENERIC_RULE_BASE_PACKAGE
  (RULE_BASES_FILENAME);

  package FACT_BASE_PACKAGE is new GENERIC_FACT_BASE_PACKAGE
  (FACT_COMPOSITE_TYPE => FACT_COMPOSITE_TYPE,
   FACT_PTR_TYPE       => FACT_PTR_TYPE,
   "<"                 => "<",
   ">"                 => ">",
   IS_EQUAL            => IS_EQUAL,
   IS_LESS_THAN        => IS_LESS_THAN,
   IS_GREATER_THAN     => IS_GREATER_THAN,
   PUT                 => PUT);

  use FACT_BASE_PACKAGE;
  use RULE_BASE_PACKAGE;
  use RULE_BASE_TYPES_PACKAGE;


  ----------------------------------------------------------------------
  --
  -- Form the Agenda base on the fact given
  --
  ----------------------------------------------------------------------
  procedure FORM
  ----------------------------------------------------------------------
  (AGENDA    : in out RULE_BASE_TYPES_PACKAGE.
                      AGENDA_LIST_PACKAGE.LIST_TYPE;
   RULE_BASE : in     STANDARD.STRING;
   FACT_PTR  : in     FACT_PTR_TYPE) is
  ----------------------------------------------------------------------

    RULE_PTR : RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;

    ANTECEDENT_PTR : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE;

    ----------------------------------------------------------------------
    --
    -- Is_rule_ready_to_fire?
    --
    ----------------------------------------------------------------------
    function IS_RULE_READY_TO_FIRE
    ----------------------------------------------------------------------
    (RULE_PTR : in RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE)
    return BOOLEAN is
```

```
  ------------------------------------------------------------------

    ANTECEDENT_PTR : RULE_BASE_TYPES_PACKAGE.
                     RULE_BASE_NODE_PTR_TYPE :=
                     RULE_PTR.CONDITIONS;

  ------------------------------------------------------------------
  begin

    while ANTECEDENT_PTR /= null
    loop

      if FACT_BASE_PACKAGE.
         IS_IN
         (ANTECEDENT_PTR.FACT, FACT_BASE_PACKAGE.FACT_BASE) then

        ANTECEDENT_PTR := ANTECEDENT_PTR.NEXT_FACT;

      else

        return FALSE;

      end if;

    end loop;

    return TRUE;

  ------------------------------------------------------------------
  end IS_RULE_READY_TO_FIRE;
  ------------------------------------------------------------------


------------------------------------------------------------------
begin -- Form

  FACT_BASE_PACKAGE.
  INSERT(FACT      => FACT_PTR,
         FACT_BASE => FACT_BASE_PACKAGE.FACT_BASE);

  RULE_BASE_PACKAGE.
  FIND
  (RULE_BASE => RULE_BASE,
   RULE_PTR  => RULE_PTR);

  while RULE_PTR /= null
  loop

    if not RULE_PTR. FIRED then

      ANTECEDENT_PTR := RULE_PTR.CONDITIONS;

      while ANTECEDENT_PTR /= null
      loop

        if IS_EQUAL(ANTECEDENT_PTR.FACT, FACT_PTR) then

          if IS_RULE_READY_TO_FIRE(RULE_PTR) then

            RULE_BASE_TYPES_PACKAGE.
```

```
                AGENDA_LIST_PACKAGE.
                PUT_ON_BACK_OF(AGENDA, RULE_PTR);

                RULE_PTR.FIRED := TRUE;
                exit;

              end if;

            end if;

            ANTECEDENT_PTR := ANTECEDENT_PTR.NEXT_FACT;

          end loop;

        end if;

        RULE_PTR := RULE_PTR.NEXT_RULE;

      end loop;
    ------------------------------------------------------------------
    end FORM;
    ------------------------------------------------------------------


    ------------------------------------------------------------------
    --
    -- Inference with the given fact
    --
    ------------------------------------------------------------------
    procedure INFERENCE
    ------------------------------------------------------------------
    (RULE_BASE : in     STANDARD.STRING;
     FACT_PTR  : in     FACT_PTR_TYPE;
     AGENDA    : in out RULE_BASE_TYPES_PACKAGE.
                        AGENDA_LIST_PACKAGE.LIST_TYPE) is
    ------------------------------------------------------------------
    begin

      FORM(AGENDA, RULE_BASE, FACT_PTR);

    ------------------------------------------------------------------
    end INFERENCE;
    ------------------------------------------------------------------


------------------------------------------------------------------
end GENERIC_RULE_BASE_INFERENCE_PACKAGE;
------------------------------------------------------------------
```

Frame Base

```
------------------------------------------------------------------------
--
-- Unit      : GENERIC_FRAME_BASE_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 3 June 1992
-- Function  : This package provides the operations to build a Frame
--             System
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     GENERIC_TREE_PACKAGE,
     GENERIC_LIST_PACKAGE;
use  TEXT_IO,
     SYSTEM_TYPES_PACKAGE;
------------------------------------------------------------------------
generic

  type SLOT_TYPE is (<>);

  type FACET_RECORD_TYPE(SLOT_KIND : SLOT_TYPE) is private;

  type FACET_RECORD_PTR_TYPE is access FACET_RECORD_TYPE;

  with function "<"
  (LEFT_FACET_PTR  : in FACET_RECORD_PTR_TYPE ;
   RIGHT_FACET_PTR : in FACET_RECORD_PTR_TYPE) return BOOLEAN;

  with function ">"
  (LEFT_FACET_PTR  : in FACET_RECORD_PTR_TYPE ;
   RIGHT_FACET_PTR : in FACET_RECORD_PTR_TYPE) return BOOLEAN;

  with function IS_EQUAL
  (FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACET_PTR  : in FACET_RECORD_PTR_TYPE) return BOOLEAN;

  with function IS_LESS_THAN
  (FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACET_PTR  : in FACET_RECORD_PTR_TYPE) return BOOLEAN;

  with function IS_GREATER_THAN
  (FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACET_PTR  : in FACET_RECORD_PTR_TYPE) return BOOLEAN;

  with procedure PUT
  (FACET_PTR : in FACET_RECORD_PTR_TYPE);

  with procedure GET
  (FACET_PTR : in out FACET_RECORD_PTR_TYPE;
   FROM_FILE : in out TEXT_IO.FILE_TYPE);

------------------------------------------------------------------------
package GENERIC_FRAME_BASE_PACKAGE is
------------------------------------------------------------------------

  type FRAME_BASE_RECORD_TYPE is private;
```

```
-- Define the facet tree

-------------------------------------------------------------------
package FACET_PACKAGE is
-------------------------------------------------------------------

  package FACET_TREE_PACKAGE is new GENERIC_TREE_PACKAGE
  (ITEM_BASE_TYPE      => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          STRING,
   ITEM_COMPOSITE_TYPE => FACET_RECORD_TYPE,
   ITEM_PTR_TYPE       => FACET_RECORD_PTR_TYPE,
   "<"                 => "<",
   ">"                 =>  ">",
   IS_EQUAL            => IS_EQUAL,
   IS_LESS_THAN        => IS_LESS_THAN,
   IS_GREATER_THAN     => IS_GREATER_THAN,
   PUT                 => PUT);

-------------------------------------------------------------------
end FACET_PACKAGE;
-------------------------------------------------------------------

use FACET_PACKAGE;

-- Define the slot tree

-------------------------------------------------------------------
package SLOT_PACKAGE is
-------------------------------------------------------------------

  type SLOT_RECORD_TYPE is
  record
    NAME           : SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     STRING;

    FACET_TREE_PTR : FACET_PACKAGE.
                     FACET_TREE_PACKAGE.
                     TREE_PTR_TYPE;
  end record;

  type SLOT_RECORD_PTR_TYPE is access SLOT_RECORD_TYPE;

  function IS_EQUAL
  (LEFT_SLOT_PTR  : in SLOT_RECORD_PTR_TYPE;
   RIGHT_SLOT_PTR : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;

  function "<"
  (LEFT_SLOT_PTR  : in SLOT_RECORD_PTR_TYPE ;
   RIGHT_SLOT_PTR : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;

  function ">"
  (LEFT_SLOT_PTR  : in SLOT_RECORD_PTR_TYPE ;
  RIGHT_SLOT_PTR : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;

  function IS_EQUAL
  (SLOT_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   SLOT_PTR  : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;
```

```
   function IS_LESS_THAN
   (SLOT_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
    SLOT_PTR  : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;

   function IS_GREATER_THAN
   (SLOT_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
    SLOT_PTR  : in SLOT_RECORD_PTR_TYPE) return BOOLEAN;

   procedure PUT
   (SLOT_PTR : in SLOT_RECORD_PTR_TYPE);

   package SLOT_TREE_PACKAGE is new GENERIC_TREE_PACKAGE
   (ITEM_BASE_TYPE      => SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING,
    ITEM_COMPOSITE_TYPE => SLOT_RECORD_TYPE,
    ITEM_PTR_TYPE       => SLOT_RECORD_PTR_TYPE,
    "<"                 => "<",
    ">"                 => ">",
    IS_EQUAL            => IS_EQUAL,
    IS_LESS_THAN        => IS_LESS_THAN,
    IS_GREATER_THAN     => IS_GREATER_THAN,
    PUT                 => PUT);

-------------------------------------------------------------------
end SLOT_PACKAGE;
-------------------------------------------------------------------
use SLOT_PACKAGE;
-------------------------------------------------------------------
package FRAME_PACKAGE is
-------------------------------------------------------------------
  type FRAME_KIND is (CLASS, INSTANCE);

  type FRAME_RECORD_TYPE(KIND : FRAME_KIND);

  type FRAME_RECORD_PTR_TYPE is access FRAME_RECORD_TYPE;

  function IS_EQUAL
  (LEFT_FRAME_PTR  : in FRAME_RECORD_PTR_TYPE;
   RIGHT_FRAME_PTR : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

  package LIST_PACKAGE is new GENERIC_LIST_PACKAGE
  (ITEM_TYPE => FRAME_RECORD_PTR_TYPE,
   IS_EQUAL  => IS_EQUAL);

  function "<"
  (LEFT_FRAME_PTR  : in FRAME_RECORD_PTR_TYPE ;
   RIGHT_FRAME_PTR : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

  function ">"
  (LEFT_FRAME_PTR  : in FRAME_RECORD_PTR_TYPE ;
   RIGHT_FRAME_PTR : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

  function IS_EQUAL
  (FRAME_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FRAME_PTR  : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

  function IS_LESS_THAN
  (FRAME_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
```

```
    FRAME_PTR  : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

  function IS_GREATER_THAN
  (FRAME_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FRAME_PTR  : in FRAME_RECORD_PTR_TYPE) return BOOLEAN;

  procedure PUT
  (FRAME_PTR : in FRAME_RECORD_PTR_TYPE);

  type FRAME_RECORD_TYPE(KIND : FRAME_KIND) is
  record
    SUPERCLASS_LIST : LIST_PACKAGE.LIST_TYPE;
    NAME            : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
    SLOT_TREE_PTR   : SLOT_PACKAGE.
                      SLOT_TREE_PACKAGE.
                      TREE_PTR_TYPE;
    case KIND is
      when CLASS    => SUBCLASS_LIST   : LIST_PACKAGE.LIST_TYPE;
      when INSTANCE => null;
    end case;
  end record;

  package FRAME_TREE_PACKAGE is new GENERIC_TREE_PACKAGE
  (ITEM_BASE_TYPE      => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          STRING,
   ITEM_COMPOSITE_TYPE => FRAME_RECORD_TYPE,
   ITEM_PTR_TYPE       => FRAME_RECORD_PTR_TYPE,
   "<"                 => "<",
   ">"                 => ">",
   IS_EQUAL            => IS_EQUAL,
   IS_LESS_THAN        => IS_LESS_THAN,
   IS_GREATER_THAN     => IS_GREATER_THAN,
   PUT                 => PUT);

-------------------------------------------------------------------
end FRAME_PACKAGE;
-------------------------------------------------------------------
use FRAME_PACKAGE;

procedure BUILD
(FRAME_BASE_RECORD          : in out FRAME_BASE_RECORD_TYPE;
 FRAME_FILE_LIST_FILENAME   : in     STANDARD.
                                     STRING);

procedure INITIALISE_FRAME_BASE
(FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE);

procedure ADD_FRAME
(FRAME_NAME        : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_TYPE        : in     FRAME_PACKAGE.
                            FRAME_KIND := FRAME_PACKAGE.
                                          CLASS;
 SUPERCLASS_NAME   : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;

 FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE);
```

```
procedure ADD_SLOT
(SLOT_NAME          : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_NAME         : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_BASE_RECORD  : in out FRAME_BASE_RECORD_TYPE);

procedure ADD_FACET
(FACET_PTR          : in     FACET_RECORD_PTR_TYPE;
 SLOT_NAME          : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_NAME         : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_BASE_RECORD  : in out FRAME_BASE_RECORD_TYPE);

procedure FIND_FACET
(FACET_NAME          : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 SLOT_NAME           : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_NAME          : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_BASE_RECORD   : in     FRAME_BASE_RECORD_TYPE;
 FACET_PTR           : in out FACET_RECORD_PTR_TYPE);

procedure FIND_SLOT
(SLOT_NAME           : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_NAME          : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_BASE_RECORD   : in     FRAME_BASE_RECORD_TYPE;
 SLOT_PTR            : in out SLOT_RECORD_PTR_TYPE);

procedure FIND_FRAME
(FRAME_NAME          : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_BASE_RECORD   : in     FRAME_BASE_RECORD_TYPE;
 FRAME_PTR           : in out FRAME_RECORD_PTR_TYPE);

procedure PUT
(FRAME_BASE_RECORD : in FRAME_BASE_RECORD_TYPE);

function IS_EMPTY
(FRAME_BASE_RECORD : in FRAME_BASE_RECORD_TYPE) return BOOLEAN;

procedure GET
(SUBFRAME_LIST     : in out SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE;
```

```
    FACET_PTR_ARRAY   : in out FACET_PACKAGE.
                               FACET_TREE_PACKAGE.
                               ITEM_ARRAY_TYPE;
    NUMBER_OF_FACETS  : in out NATURAL;
    SLOT_NAME         : in     SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
    FRAME_PTR         : in     FRAME_RECORD_PTR_TYPE);

   procedure GET
   (SUBFRAME_LIST     : in out FRAME_PACKAGE.
                               LIST_PACKAGE.
                               LIST_TYPE;
    FRAME_NAME        : in     SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
    FRAME_BASE_RECORD : in     FRAME_BASE_RECORD_TYPE);

   procedure CLEAR
   (FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE);

   procedure ADD_FRAME
   (FRAME_PTR         : in     FRAME_PACKAGE.FRAME_RECORD_PTR_TYPE;
    SUPERCLASS_NAME   : in     SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
    FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE);

   procedure DELETE_FRAME
   (FRAME_PTR         : in     FRAME_PACKAGE.FRAME_RECORD_PTR_TYPE;
    SUPERCLASS_NAME   : in     SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
    FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE);

private

  type FRAME_BASE_RECORD_TYPE is
  record

    FRAME_BASE_PTR           : FRAME_PACKAGE.
                               FRAME_RECORD_PTR_TYPE;
    FRAME_SEARCH_TREE_PTR    : FRAME_PACKAGE.
                               FRAME_TREE_PACKAGE.
                               TREE_PTR_TYPE;

  end record;

-------------------------------------------------------------------------
end GENERIC_FRAME_BASE_PACKAGE;
-------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unit      : GENERIC_FRAME_BASE_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 4 June 1992
-- Function  : This package provides the operations to build a Frame
--             System
--
------------------------------------------------------------------
package body GENERIC_FRAME_BASE_PACKAGE is
------------------------------------------------------------------

  ------------------------------------------------------------------
  package body SLOT_PACKAGE is
  ------------------------------------------------------------------

      -- Operations for the instantiation of the slot tree

    ------------------------------------------------------------------
    function IS_EQUAL
    ------------------------------------------------------------------
    (LEFT_SLOT_PTR  : in SLOT_RECORD_PTR_TYPE;
     RIGHT_SLOT_PTR : in SLOT_RECORD_PTR_TYPE) return BOOLEAN is
    ------------------------------------------------------------------
    begin
      return SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             IS_EQUAL(LEFT  => LEFT_SLOT_PTR.NAME,
                      RIGHT => RIGHT_SLOT_PTR.NAME);
    ------------------------------------------------------------------
    end IS_EQUAL;
    ------------------------------------------------------------------


    ------------------------------------------------------------------
    function "<"
    ------------------------------------------------------------------
  (LEFT_SLOT_PTR  : in SLOT_RECORD_PTR_TYPE ;
   RIGHT_SLOT_PTR : in SLOT_RECORD_PTR_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_LESS_THAN(LEFT  => LEFT_SLOT_PTR.NAME,
                        RIGHT => RIGHT_SLOT_PTR.NAME);
  ------------------------------------------------------------------
  end "<";
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function ">"
  ------------------------------------------------------------------
  (LEFT_SLOT_PTR  : in SLOT_RECORD_PTR_TYPE ;
   RIGHT_SLOT_PTR : in SLOT_RECORD_PTR_TYPE) return BOOLEAN is
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_GREATER_THAN(LEFT  => LEFT_SLOT_PTR.NAME,
```

```
                              RIGHT => RIGHT_SLOT_PTR.NAME);
-------------------------------------------------------------------
end ">";
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(SLOT_NAME : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 SLOT_PTR  : in SLOT_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL(LEFT  => SLOT_NAME,
                  RIGHT => SLOT_PTR.NAME);
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_LESS_THAN
-------------------------------------------------------------------
(SLOT_NAME : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 SLOT_PTR  : in SLOT_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_LESS_THAN(LEFT  => SLOT_NAME,
                      RIGHT => SLOT_PTR.NAME);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_GREATER_THAN
-------------------------------------------------------------------
(SLOT_NAME : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 SLOT_PTR  : in SLOT_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_GREATER_THAN(LEFT  => SLOT_NAME,
                         RIGHT => SLOT_PTR.NAME);
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------
```

```
  ------------------------------------------------------------------
  procedure PUT
  ------------------------------------------------------------------
  (SLOT_PTR : in SLOT_RECORD_PTR_TYPE) is
  ------------------------------------------------------------------
  begin

    FACET_PACKAGE.
    FACET_TREE_PACKAGE.
    PRINT
    (TREE_PTR => SLOT_PTR.FACET_TREE_PTR);

  ------------------------------------------------------------------
  end PUT;
  ------------------------------------------------------------------


------------------------------------------------------------------
end SLOT_PACKAGE;
------------------------------------------------------------------


------------------------------------------------------------------
package body FRAME_PACKAGE is
------------------------------------------------------------------

    -- Operations for the instantiation of the frame search tree

  ------------------------------------------------------------------
  function IS_EQUAL
  ------------------------------------------------------------------
  (LEFT_FRAME_PTR  : in FRAME_RECORD_PTR_TYPE;
   RIGHT_FRAME_PTR : in FRAME_RECORD_PTR_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_EQUAL(LEFT  => LEFT_FRAME_PTR.NAME,
                    RIGHT => RIGHT_FRAME_PTR.NAME);
  ------------------------------------------------------------------
  end IS_EQUAL;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function "<"
  ------------------------------------------------------------------
  (LEFT_FRAME_PTR  : in FRAME_RECORD_PTR_TYPE ;
   RIGHT_FRAME_PTR : in FRAME_RECORD_PTR_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_LESS_THAN(LEFT  => LEFT_FRAME_PTR.NAME,
                        RIGHT => RIGHT_FRAME_PTR.NAME);
  ------------------------------------------------------------------
  end "<";
  ------------------------------------------------------------------
```

```
-------------------------------------------------------------------
function ">"
-------------------------------------------------------------------
(LEFT_FRAME_PTR  : in FRAME_RECORD_PTR_TYPE ;
 RIGHT_FRAME_PTR : in FRAME_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_GREATER_THAN(LEFT  => LEFT_FRAME_PTR.NAME,
                         RIGHT => RIGHT_FRAME_PTR.NAME);
-------------------------------------------------------------------
end ">";
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(FRAME_NAME : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 FRAME_PTR  : in FRAME_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL(LEFT  => FRAME_NAME,
                  RIGHT => FRAME_PTR.NAME);
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_LESS_THAN
-------------------------------------------------------------------
(FRAME_NAME : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 FRAME_PTR  : in FRAME_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_LESS_THAN(LEFT  => FRAME_NAME,
                      RIGHT => FRAME_PTR.NAME);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_GREATER_THAN
-------------------------------------------------------------------
(FRAME_NAME : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 FRAME_PTR  : in FRAME_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
```

```
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_GREATER_THAN(LEFT  => FRAME_NAME,
                           RIGHT => FRAME_PTR.NAME);
  ------------------------------------------------------------------
  end IS_GREATER_THAN;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure PUT
  ------------------------------------------------------------------
  (FRAME_PTR : in FRAME_RECORD_PTR_TYPE) is
  ------------------------------------------------------------------
  begin

    SLOT_PACKAGE.
    SLOT_TREE_PACKAGE.
    PRINT
    (TREE_PTR => FRAME_PTR.SLOT_TREE_PTR);

  ------------------------------------------------------------------
  end PUT;
  ------------------------------------------------------------------


------------------------------------------------------------------
end FRAME_PACKAGE;
------------------------------------------------------------------

  -- Operations on the frame-base

  ------------------------------------------------------------------

  ------------------------------------------------------------------
  procedure ATTACH
  ------------------------------------------------------------------
  (FRAME_PTR          : in      FRAME_PACKAGE.
                                FRAME_RECORD_PTR_TYPE;
   TO_SUPERCLASS_PTR : in  out FRAME_PACKAGE.
                                FRAME_RECORD_PTR_TYPE) is
  ------------------------------------------------------------------
  begin

    FRAME_PACKAGE.
    LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST      => TO_SUPERCLASS_PTR.
                  SUBCLASS_LIST,
     THIS_ITEM => FRAME_PTR);

    FRAME_PACKAGE.
    LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST      => FRAME_PTR.
                  SUPERCLASS_LIST,
     THIS_ITEM => TO_SUPERCLASS_PTR);

  ------------------------------------------------------------------
```

```
end ATTACH;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADD_FRAME
-------------------------------------------------------------------
(FRAME_NAME         : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_TYPE         : in     FRAME_PACKAGE.
                             FRAME_KIND := FRAME_PACKAGE.
                                           CLASS;
 SUPERCLASS_NAME    : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is
  -----------------------------------------------------------------

  NEW_FRAME_PTR : FRAME_PACKAGE.
                  FRAME_RECORD_PTR_TYPE := new FRAME_PACKAGE.
                                               FRAME_RECORD_TYPE
                                               (FRAME_TYPE);

  SUPERCLASS_PTR : FRAME_PACKAGE.
                   FRAME_RECORD_PTR_TYPE;

-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_STRING => FRAME_NAME,
   TO_THE_STRING   => NEW_FRAME_PTR.
                      NAME);

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
  (ITEM        => SUPERCLASS_NAME,
   TREE_PTR    => FRAME_BASE_RECORD.
                  FRAME_SEARCH_TREE_PTR,
   RESULT_PTR  => SUPERCLASS_PTR);

  ATTACH
  (FRAME_PTR          => NEW_FRAME_PTR,
   TO_SUPERCLASS_PTR => SUPERCLASS_PTR);

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  INSERT
  (ITEM_PTR => NEW_FRAME_PTR,
   TREE_PTR => FRAME_BASE_RECORD.
               FRAME_SEARCH_TREE_PTR);

-------------------------------------------------------------------
end ADD_FRAME;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure ADD_SLOT
-------------------------------------------------------------------
(SLOT_NAME         : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_NAME        : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------

  NEW_SLOT_PTR : SLOT_PACKAGE.
                 SLOT_RECORD_PTR_TYPE := new SLOT_PACKAGE.
                                             SLOT_RECORD_TYPE;

  FRAME_PTR : FRAME_PACKAGE.
              FRAME_RECORD_PTR_TYPE;

-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_STRING => SLOT_NAME,
   TO_THE_STRING   => NEW_SLOT_PTR.
                      NAME);

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
  (ITEM         => FRAME_NAME,
   TREE_PTR     => FRAME_BASE_RECORD.
                   FRAME_SEARCH_TREE_PTR,
   RESULT_PTR   => FRAME_PTR);

  SLOT_PACKAGE.
  SLOT_TREE_PACKAGE.
  INSERT
  (ITEM_PTR => NEW_SLOT_PTR,
   TREE_PTR => FRAME_PTR.SLOT_TREE_PTR);

-------------------------------------------------------------------
end ADD_SLOT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADD_FACET
-------------------------------------------------------------------
(FACET_PTR         : in     FACET_RECORD_PTR_TYPE;
 SLOT_NAME         : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_NAME        : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is
```

```
------------------------------------------------------------------

  FRAME_PTR : FRAME_PACKAGE.
              FRAME_RECORD_PTR_TYPE;

  SLOT_PTR : SLOT_PACKAGE.
             SLOT_RECORD_PTR_TYPE;

------------------------------------------------------------------
begin

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
  (ITEM         => FRAME_NAME,
   TREE_PTR     => FRAME_BASE_RECORD.
                   FRAME_SEARCH_TREE_PTR,
   RESULT_PTR   => FRAME_PTR);

  SLOT_PACKAGE.
  SLOT_TREE_PACKAGE.
  FIND
  (ITEM         => SLOT_NAME,
   TREE_PTR     => FRAME_PTR.
                   SLOT_TREE_PTR,
   RESULT_PTR   => SLOT_PTR);


  FACET_PACKAGE.
  FACET_TREE_PACKAGE.
  INSERT
  (ITEM_PTR => FACET_PTR,
   TREE_PTR => SLOT_PTR.FACET_TREE_PTR);

------------------------------------------------------------------
end ADD_FACET;
------------------------------------------------------------------


------------------------------------------------------------------
procedure FIND_FACET
------------------------------------------------------------------
(FACET_NAME          : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 SLOT_NAME           : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_NAME          : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 FRAME_BASE_RECORD   : in     FRAME_BASE_RECORD_TYPE;
 FACET_PTR           : in out FACET_RECORD_PTR_TYPE) is
------------------------------------------------------------------

  SLOT_PTR : SLOT_PACKAGE.
             SLOT_RECORD_PTR_TYPE;

  FRAME_PTR : FRAME_PACKAGE.
              FRAME_RECORD_PTR_TYPE;
```

```
-------------------------------------------------------------------
begin

  -- Find the frame

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
  (ITEM        => FRAME_NAME,
   TREE_PTR    => FRAME_BASE_RECORD.
                  FRAME_SEARCH_TREE_PTR,
   RESULT_PTR => FRAME_PTR);

  -- Find the slot

  SLOT_PACKAGE.
  SLOT_TREE_PACKAGE.
  FIND
  (ITEM        => SLOT_NAME,
   TREE_PTR    => FRAME_PTR.
                  SLOT_TREE_PTR,
   RESULT_PTR => SLOT_PTR);

  -- Find the facet

  FACET_PACKAGE.
  FACET_TREE_PACKAGE.
  FIND
  (ITEM        => FACET_NAME,
   TREE_PTR    => SLOT_PTR.
                  FACET_TREE_PTR,
   RESULT_PTR => FACET_PTR);

-------------------------------------------------------------------
end FIND_FACET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure FIND_SLOT
-------------------------------------------------------------------
(SLOT_NAME          : in    SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_NAME         : in    SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_BASE_RECORD  : in    FRAME_BASE_RECORD_TYPE;
 SLOT_PTR           : in out SLOT_RECORD_PTR_TYPE) is
-------------------------------------------------------------------

  FRAME_PTR : FRAME_PACKAGE.
              FRAME_RECORD_PTR_TYPE;

-------------------------------------------------------------------
begin

  -- Find the frame
```

```
   FRAME_PACKAGE.
   FRAME_TREE_PACKAGE.
   FIND
   (ITEM        => FRAME_NAME,
    TREE_PTR    => FRAME_BASE_RECORD.
                   FRAME_SEARCH_TREE_PTR,
    RESULT_PTR => FRAME_PTR);

   -- Find the slot

   SLOT_PACKAGE.
   SLOT_TREE_PACKAGE.
   FIND
   (ITEM        => SLOT_NAME,
    TREE_PTR    => FRAME_PTR.
                   SLOT_TREE_PTR,
    RESULT_PTR => SLOT_PTR);

-------------------------------------------------------------------
end FIND_SLOT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure FIND_FRAME
-------------------------------------------------------------------
(FRAME_NAME         : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_BASE_RECORD : in     FRAME_BASE_RECORD_TYPE;
 FRAME_PTR          : in out FRAME_RECORD_PTR_TYPE) is
-------------------------------------------------------------------
begin

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
  (ITEM        => FRAME_NAME,
   TREE_PTR    => FRAME_BASE_RECORD.
                  FRAME_SEARCH_TREE_PTR,
   RESULT_PTR => FRAME_PTR);
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception in GENERIC_FRAME_BASE_PACKAGE at " &
     "FIND_FRAME");

-------------------------------------------------------------------
end FIND_FRAME;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET_ATTRIBUTES_OF
-------------------------------------------------------------------
(FRAME_NAME : in     SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     STRING;
 FRAME_TREE : in     FRAME_PACKAGE.
```

```
                        FRAME_TREE_PACKAGE.
                        TREE_PTR_TYPE;
   ON_LIST     : in out FRAME_PACKAGE.
                        LIST_PACKAGE.
                        LIST_TYPE) is
  ------------------------------------------------------------------
  begin
    null;
  ------------------------------------------------------------------
  end GET_ATTRIBUTES_OF;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  --
  --  Initilise_frame_base adds a root frame to an empty system
  --
  ------------------------------------------------------------------
  procedure INITIALISE_FRAME_BASE
  (FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is

    FRAME_PTR : FRAME_PACKAGE.
                FRAME_RECORD_PTR_TYPE := new FRAME_PACKAGE.
                                             FRAME_RECORD_TYPE
                                             (FRAME_PACKAGE.CLASS);

  begin

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_SUBSTRING => "ROOT",
       TO_THE_STRING      => FRAME_PTR.NAME);

      FRAME_BASE_RECORD.FRAME_BASE_PTR := FRAME_PTR;

      FRAME_PACKAGE.
      FRAME_TREE_PACKAGE.
      INSERT
      (ITEM_PTR => FRAME_BASE_RECORD.FRAME_BASE_PTR,
       TREE_PTR => FRAME_BASE_RECORD.FRAME_SEARCH_TREE_PTR);
  ------------------------------------------------------------------
  exception
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in GENERIC_FRAME_BASE_PACKAGE at " &
       "INITIALISE_BLACKBOARD");
  ------------------------------------------------------------------
  end INITIALISE_FRAME_BASE;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure BUILD
  ------------------------------------------------------------------
  (FRAME_BASE_RECORD        : in out FRAME_BASE_RECORD_TYPE;
   FRAME_FILE_LIST_FILENAME : in     STANDARD.
                                     STRING) is
  ------------------------------------------------------------------
```

```
   subtype STRING_LENGTH is POSITIVE range 1 .. 80;
   type INPUT_TOKEN_TYPE is
        (CLASS, INSTANCE, SUPERCLASS, SLOT, FACET);

   FRAME_FILE_LIST_FILE : TEXT_IO.FILE_TYPE;
   FRAME_FILE           : TEXT_IO.FILE_TYPE;
   FRAME_FILE_FILENAME  : STANDARD.STRING(STRING_LENGTH);

   INPUT_TOKEN,
   FRAME_NAME,
   SUPERCLASS_NAME,
   SLOT_NAME : STANDARD.STRING(STRING_LENGTH);

   CURRENT_SUPERCLASS_PTR : FRAME_PACKAGE.
                            FRAME_RECORD_PTR_TYPE;
   CURRENT_FRAME_PTR      : FRAME_PACKAGE.
                            FRAME_RECORD_PTR_TYPE;
   CURRENT_SLOT_PTR       : SLOT_PACKAGE.
                            SLOT_RECORD_PTR_TYPE;
   CURRENT_FACET_PTR      : FACET_RECORD_PTR_TYPE;


   TOKEN_LAST                : STRING_LENGTH;
   FRAME_FILE_FILENAME_LAST  : STRING_LENGTH;

   SYNTAX_ERROR : exception;
-------------------------------------------------------------------
begin
   ----------------------------------------------------------------

     -- Establish the frame-base root

     CURRENT_FRAME_PTR := new FRAME_PACKAGE.
                              FRAME_RECORD_TYPE
                              (FRAME_PACKAGE.CLASS);

     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     COPY
     (FROM_THE_SUBSTRING => "ROOT",
      TO_THE_STRING      => CURRENT_FRAME_PTR.NAME);

     FRAME_BASE_RECORD.FRAME_BASE_PTR := CURRENT_FRAME_PTR;

     FRAME_PACKAGE.
     FRAME_TREE_PACKAGE.
     INSERT
     (ITEM_PTR => FRAME_BASE_RECORD.FRAME_BASE_PTR,
      TREE_PTR => FRAME_BASE_RECORD.FRAME_SEARCH_TREE_PTR);

   ----------------------------------------------------------------

   TEXT_IO.
   OPEN
   (FILE => FRAME_FILE_LIST_FILE,
    MODE => TEXT_IO.IN_FILE,
    NAME => FRAME_FILE_LIST_FILENAME);

   while not TEXT_IO.END_OF_FILE(FRAME_FILE_LIST_FILE)
   loop
```

```
TEXT_IO.
GET_LINE
(FILE => FRAME_FILE_LIST_FILE,
 ITEM => FRAME_FILE_FILENAME,
 LAST => FRAME_FILE_FILENAME_LAST);

TEXT_IO.OPEN(FILE => FRAME_FILE,
             MODE => TEXT_IO.IN_FILE,
             NAME => FRAME_FILE_FILENAME
                     (1..FRAME_FILE_FILENAME_LAST));

-- Get the first frame type

TEXT_IO.GET_LINE(FILE => FRAME_FILE,
                 ITEM => INPUT_TOKEN,
                 LAST => TOKEN_LAST);

while not TEXT_IO.END_OF_FILE(FRAME_FILE)  -- Get all frames
loop
  if FRAME_PACKAGE.FRAME_KIND'VALUE
     (INPUT_TOKEN(1 .. TOKEN_LAST)) =

         FRAME_PACKAGE.
         CLASS   then

     CURRENT_FRAME_PTR := new FRAME_PACKAGE.
                              FRAME_RECORD_TYPE
                              (FRAME_PACKAGE.CLASS);

  else

      CURRENT_FRAME_PTR := new FRAME_PACKAGE.
                               FRAME_RECORD_TYPE
                               (FRAME_PACKAGE.INSTANCE);

  end if;

  -- Get the frame name

  TEXT_IO.
  GET_LINE(FILE => FRAME_FILE,
           ITEM => FRAME_NAME,
           LAST => TOKEN_LAST);

  -- Store it in the frame

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY(FROM_THE_SUBSTRING => FRAME_NAME(1..TOKEN_LAST),
       TO_THE_STRING      => CURRENT_FRAME_PTR.NAME);

  -- What is the next token

  TEXT_IO.GET_LINE(FILE => FRAME_FILE,
                   ITEM => INPUT_TOKEN,
                   LAST => TOKEN_LAST);

  -- If it is Superclass, slot or facet then process it
```

```
        TEXT_IO.
        GET_LINE
        (FILE => FRAME_FILE,
         ITEM => SLOT_NAME,
         LAST => TOKEN_LAST);

        -- Store name in the frame

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        COPY
        (FROM_THE_SUBSTRING => SLOT_NAME(1..TOKEN_LAST),
         TO_THE_STRING      => CURRENT_SLOT_PTR.NAME);

        -- Add slot to slot tree of current frame

        SLOT_PACKAGE.
        SLOT_TREE_PACKAGE.
        INSERT
        (ITEM_PTR => CURRENT_SLOT_PTR,
         TREE_PTR => CURRENT_FRAME_PTR.
                     SLOT_TREE_PTR);

      elsif INPUT_TOKEN_TYPE'
            VALUE(INPUT_TOKEN(1 .. TOKEN_LAST)) = FACET then

      CURRENT_FACET_PTR := new FACET_RECORD_TYPE
                                (SLOT_TYPE'
                                 VALUE
                                 (SYSTEM_TYPES_PACKAGE.
                                  DYNAMIC_STRING.
                                  SUBSTRING_OF
                                  (THE_STRING =>
                                   CURRENT_SLOT_PTR.
                                   NAME)));

        -- Get facet from application

        GET
        (FACET_PTR => CURRENT_FACET_PTR,
         FROM_FILE => FRAME_FILE);

         -- Add new facet to the current slot facet tree

         FACET_PACKAGE.
         FACET_TREE_PACKAGE.
         INSERT
         (ITEM_PTR => CURRENT_FACET_PTR,
          TREE_PTR => CURRENT_SLOT_PTR.
                      FACET_TREE_PTR);

      else raise SYNTAX_ERROR;

    end if;

    exit when TEXT_IO.END_OF_FILE(FRAME_FILE);

    -- See what next token is

    TEXT_IO.
```

```
while INPUT_TOKEN_TYPE'VALUE
      (INPUT_TOKEN(1..TOKEN_LAST)) in SUPERCLASS .. FACET
loop
  if INPUT_TOKEN_TYPE'VALUE(INPUT_TOKEN(1 .. TOKEN_LAST)) =
     SUPERCLASS then -- Get the superclass name

     TEXT_IO.
     GET_LINE
     (FILE => FRAME_FILE,
      ITEM => SUPERCLASS_NAME,
      LAST => TOKEN_LAST);

     declare

       TEMP_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
     begin

       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       COPY
       (FROM_THE_SUBSTRING => SUPERCLASS_NAME
                              (1 .. TOKEN_LAST),
        TO_THE_STRING      => TEMP_DYNAMIC_STRING);

       -- Find the superclass in the frame-base

       FRAME_PACKAGE.
       FRAME_TREE_PACKAGE.
       FIND
       (ITEM        => TEMP_DYNAMIC_STRING,
        TREE_PTR    => FRAME_BASE_RECORD.
                       FRAME_SEARCH_TREE_PTR,
        RESULT_PTR => CURRENT_SUPERCLASS_PTR);

     end;

     -- Attach the current frame to its superclass

     ATTACH
     (FRAME_PTR          => CURRENT_FRAME_PTR,
      TO_SUPERCLASS_PTR => CURRENT_SUPERCLASS_PTR);

     -- Add the current frame to the search tree

     FRAME_PACKAGE.
     FRAME_TREE_PACKAGE.
     INSERT
     (ITEM_PTR => CURRENT_FRAME_PTR,
      TREE_PTR => FRAME_BASE_RECORD.
      FRAME_SEARCH_TREE_PTR);

   elsif INPUT_TOKEN_TYPE'
         VALUE(INPUT_TOKEN(1 .. TOKEN_LAST)) = SLOT then
     CURRENT_SLOT_PTR := new SLOT_PACKAGE.
                         SLOT_RECORD_TYPE;

     -- Get the slot name
```

```
        GET_LINE
        (FILE => FRAME_FILE,
         ITEM => INPUT_TOKEN,
         LAST => TOKEN_LAST);

      end loop;

    end loop;

    TEXT_IO.CLOSE(FILE => FRAME_FILE);

  end loop;

  TEXT_IO.CLOSE(FRAME_FILE_LIST_FILE);
-------------------------------------------------------------------
exception
  when SYNTAX_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception raised in GENERIC_FRAME_PACKAGE at BUILD");
-------------------------------------------------------------------
end BUILD;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PUT(FRAME_BASE_RECORD : in FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------

  procedure PRINT
  (FRAME_BASE_PTR : in FRAME_PACKAGE.
                    FRAME_RECORD_PTR_TYPE) is

    TEMP_PTR : FRAME_PACKAGE.
               FRAME_RECORD_PTR_TYPE;

  begin

    TEXT_IO.SET_COL(1);

    if FRAME_BASE_PTR /= null then

      -- Print the frame name plus the slot and facet trees.

      if not SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             IS_EQUAL
             (LEFT  => "ROOT",
              RIGHT => FRAME_BASE_PTR.NAME) then

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        PUT
        (THE_STRING => FRAME_BASE_PTR.NAME);
        TEXT_IO.NEW_LINE;

      end if;

      if FRAME_BASE_PTR.KIND = CLASS then

        for NODE in 1 .. FRAME_PACKAGE.
```

```
                        LIST_PACKAGE.
                        LENGTH_OF
                        (LIST =>FRAME_BASE_PTR.
                                SUBCLASS_LIST)
        loop

           PRINT
           (FRAME_BASE_PTR => FRAME_PACKAGE.
                              LIST_PACKAGE.
                              ITEM_AT
                              (LIST        => FRAME_BASE_PTR.
                                              SUBCLASS_LIST,
                               NODE_NUMBER => NODE));

          end loop;

      end if;

    end if;

      FRAME_PACKAGE.
      PUT
      (FRAME_BASE_PTR);
-------------------------------------------------------------------
  end PRINT;
-------------------------------------------------------------------
begin

  PRINT(FRAME_BASE_PTR => FRAME_BASE_RECORD.FRAME_BASE_PTR);

-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(SUBFRAME_LIST     : in out SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE;
 FACET_PTR_ARRAY   : in out FACET_PACKAGE.
                            FACET_TREE_PACKAGE.
                            ITEM_ARRAY_TYPE;
 NUMBER_OF_FACETS  : in out NATURAL;
 SLOT_NAME         : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_PTR         : in     FRAME_RECORD_PTR_TYPE) is
-------------------------------------------------------------------

  SLOT_PTR    : SLOT_RECORD_PTR_TYPE;

  -----------------------------------------------------------------
  procedure GET_FRAME_LIST
  -----------------------------------------------------------------
  (FRAME_PTR      : in     FRAME_PACKAGE.
                           FRAME_RECORD_PTR_TYPE;
   SUBFRAME_LIST : in out SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
```

```
                              LIST_TYPE) is

    FRAME_NAME : SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
  ------------------------------------------------------------------
  begin

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_STRING => FRAME_PTR.
                          NAME,
       TO_THE_STRING   => FRAME_NAME);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => SUBFRAME_LIST,
       THIS_ITEM => FRAME_NAME);

    for COUNT in 1 .. FRAME_PACKAGE.
                      LIST_PACKAGE.
                      LENGTH_OF
                      (LIST => FRAME_PTR.
                               SUBCLASS_LIST)
    loop

      GET_FRAME_LIST(FRAME_PTR => FRAME_PACKAGE.
                                  LIST_PACKAGE.
                                  ITEM_AT
                                  (LIST        => FRAME_PTR.
                                                  SUBCLASS_LIST,
                                   NODE_NUMBER =>'COUNT),
                     SUBFRAME_LIST => SUBFRAME_LIST);
    end loop;
  ------------------------------------------------------------------
  end GET_FRAME_LIST;
------------------------------------------------------------------
begin

  GET_FRAME_LIST(FRAME_PTR     => FRAME_PTR,
                 SUBFRAME_LIST => SUBFRAME_LIST);

  SLOT_PACKAGE.
  SLOT_TREE_PACKAGE.
  FIND
  (ITEM       => SLOT_NAME,
   TREE_PTR   => FRAME_PTR.
                 SLOT_TREE_PTR,
   RESULT_PTR => SLOT_PTR);

  if SLOT_PTR = null then

    NUMBER_OF_FACETS := 0;

  else

    FACET_PACKAGE.
    FACET_TREE_PACKAGE.
```

```
    FORM_ARRAY
    (TREE_PTR     => SLOT_PTR.FACET_TREE_PTR,
     ITEM_ARRAY   => FACET_PTR_ARRAY,
     ITEM_NUMBER => NUMBER_OF_FACETS);

  end if;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in GENERIC_FRAME_BASE_PACKAGE at" &
     "GET subframe name list and facet array");
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(SUBFRAME_LIST     : in out FRAME_PACKAGE.
                            LIST_PACKAGE.
                            LIST_TYPE;
 FRAME_NAME         : in    SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
 FRAME_BASE_RECORD : in    FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
  FRAME_PTR : FRAME_RECORD_PTR_TYPE;
  SUBFRAME_PTR : FRAME_RECORD_PTR_TYPE;
-------------------------------------------------------------------
begin

  FIND_FRAME
  (FRAME_NAME         => FRAME_NAME,
   FRAME_BASE_RECORD => FRAME_BASE_RECORD,
   FRAME_PTR          => FRAME_PTR);

  for SUBFRAME_NUMBER in 1 .. LIST_PACKAGE.
                              LENGTH_OF
                              (LIST => FRAME_PTR.
                                       SUBCLASS_LIST)
  loop

    LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST => SUBFRAME_LIST,
     THIS_ITEM => LIST_PACKAGE.
                  ITEM_AT
                  (LIST          => FRAME_PTR.
                                    SUBCLASS_LIST,
                   NODE_NUMBER =>  SUBFRAME_NUMBER));

  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
  TEXT_IO.PUT_LINE
    ("Exception OTHERS in GENERIC_FRAME_BASED_PACKAGE at " &
     "GET subframe list");
```

```
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_EMPTY
-------------------------------------------------------------------
(FRAME_BASE_RECORD : in FRAME_BASE_RECORD_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return FRAME_PACKAGE.
         LIST_PACKAGE.
         IS_EMPTY
         (FRAME_BASE_RECORD.
          FRAME_BASE_PTR.
          SUBCLASS_LIST);
-------------------------------------------------------------------
end IS_EMPTY;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure CLEAR
-------------------------------------------------------------------
(FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
begin

  -- Need garbage collector here!

  FRAME_BASE_RECORD.FRAME_BASE_PTR := null;
  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  CLEAR
  (FRAME_BASE_RECORD.FRAME_SEARCH_TREE_PTR);

  -- Re-initialise

  INITIALISE_FRAME_BASE(FRAME_BASE_RECORD => FRAME_BASE_RECORD);
-------------------------------------------------------------------
end CLEAR;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADD_FRAME
-------------------------------------------------------------------
(FRAME_PTR          : in     FRAME_PACKAGE.FRAME_RECORD_PTR_TYPE;
 SUPERCLASS_NAME    : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
  SUPERCLASS_PTR : FRAME_RECORD_PTR_TYPE;
-------------------------------------------------------------------
begin
  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
```

```
   (ITEM       => SUPERCLASS_NAME,
    TREE_PTR   => FRAME_BASE_RECORD.
                  FRAME_SEARCH_TREE_PTR,
    RESULT_PTR => SUPERCLASS_PTR);

  ATTACH
   (FRAME_PTR          => FRAME_PTR,
    TO_SUPERCLASS_PTR => SUPERCLASS_PTR);

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  INSERT
  (ITEM_PTR => FRAME_PTR,
   TREE_PTR => FRAME_BASE_RECORD.
               FRAME_SEARCH_TREE_PTR);
-------------------------------------------------------------------
end ADD_FRAME;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure DELETE_FRAME
-------------------------------------------------------------------
(FRAME_PTR          : in     FRAME_PACKAGE.FRAME_RECORD_PTR_TYPE;
 SUPERCLASS_NAME    : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 FRAME_BASE_RECORD : in out FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
  SUPERCLASS_PTR : FRAME_RECORD_PTR_TYPE;
-------------------------------------------------------------------
begin
  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  FIND
  (ITEM       => SUPERCLASS_NAME,
   TREE_PTR   => FRAME_BASE_RECORD.
                 FRAME_SEARCH_TREE_PTR,
   RESULT_PTR => SUPERCLASS_PTR);

  FRAME_PACKAGE.
  LIST_PACKAGE.
  DELETE_FROM
  (LIST      => SUPERCLASS_PTR.SUBCLASS_LIST,
   THIS_ITEM => FRAME_PTR);

  FRAME_PACKAGE.
  LIST_PACKAGE.
  DELETE_FROM
  (LIST      => FRAME_PTR.SUPERCLASS_LIST,
   THIS_ITEM => SUPERCLASS_PTR);

  FRAME_PACKAGE.
  FRAME_TREE_PACKAGE.
  DELETE
  (ITEM_PTR => FRAME_PTR,
   TREE_PTR => FRAME_BASE_RECORD.
               FRAME_SEARCH_TREE_PTR);

-------------------------------------------------------------------
```

```
  end DELETE_FRAME;
  ------------------------------------------------------------------------

------------------------------------------------------------------------
end GENERIC_FRAME_BASE_PACKAGE;
------------------------------------------------------------------------
```

The Integration of Multiple and Diverse Knowledge Representation Paradigms using a Blackboard Architecture

Annex B

Integrating Abstract Knowledge Types

A University Timetable System

A Thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy in Computer Science

Alan Harrison BA MSc

Faculty of Mathematics and Computing
The Open University

December 1994

Contents

A University Timetabling System

Timetable Knowledge

A University Timetabling System

Timetable Types

```
-------------------------------------------------------------------------
--
-- Unit        : TIMETABLE_TYPES_PACKAGE specification
-- Author      : A Harrison Software Engineering Group,
--               Cranfield University, RMCS, Shrivenam
-- Date        : 21 December 1991
-- Function    : This package provides the TIMETABLE application
--               definitions
--
-------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     RULE_BASE_TYPES_PACKAGE,
     GENERIC_LIST_PACKAGE,
     GENERIC_TREE_PACKAGE,
     GENERIC_FRAME_BASE_PACKAGE;
use  TEXT_IO;
-------------------------------------------------------------------------
package TIMETABLE_TYPES_PACKAGE is
-------------------------------------------------------------------------
  package INTEGER_TEXT_IO is new TEXT_IO.INTEGER_IO(INTEGER);
  -----------------------------------------------------------------------

  -- Vertical blackboard divisions

  type TIMETABLE_LEVEL_TYPE is
  (DAYS,
   DEGREE_ACTIVITIES,
   COMMON_MODULE_REQUIREMENTS,
   MODULE_REQUIREMENTS,
   DEGREE_MODULES,
   EVENT_LISTS);

  -- Horizontal blackboard divisions

  type TIMETABLE_ITEM_TYPE is
  (MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY,
   LECTURES, TUTORIALS, CAROUSEL, PRACTICALS,
   COMMON_MODULE_REQUIREMENT_LIST,
   ESE,
   IT,
   COMMON_EVENTS,
   PERIOD_EVENTS);

  -- Blackboard nodes

  type TIMETABLE_NODE_KIND_TYPE is
  (PERIOD_KIND,
   MODULE_ACTIVITY_KIND,
   MODULE_COMMON_REQUIREMENT_KIND,
   MODULE_REQUIREMENT_KIND,
   MODULE_KIND);
```

```
type TIMETABLE_NODE_RECORD
(KIND : TIMETABLE_NODE_KIND_TYPE := MODULE_ACTIVITY_KIND);

type TIMETABLE_NODE_PTR_TYPE is access TIMETABLE_NODE_RECORD;

-- Module activities

type ACTIVITY_TYPE is
(LECTURE,
 TUTORIAL,
 CAROUSEL,
 PRACTICAL);

-- Periods in a day

NUMBER_OF_PERIODS : constant := 7;

-- Weeks in a term

NUMBER_OF_WEEKS : constant := 11;

type MODULE_REQUIREMENT_TYPE is array(ACTIVITY_TYPE) of BOOLEAN;

subtype PERIOD_NUMBER_TYPE is POSITIVE
                              range 1 .. NUMBER_OF_PERIODS;

subtype WEEK_NUMBER_TYPE is POSITIVE range 1 .. NUMBER_OF_WEEKS;

type WEEK_ARRAY_TYPE is array(WEEK_NUMBER_TYPE) of BOOLEAN;

type DAY_TYPE is (mon, tue, wed, thu, fri);

FREE : WEEK_ARRAY_TYPE := (WEEK_NUMBER_TYPE => FALSE);
BUSY : WEEK_ARRAY_TYPE := (WEEK_NUMBER_TYPE => TRUE);

-----------------------------------------------------------------

-- Each period is a frame base with the facet storing the detail
-- about a single period

type PERIOD_SLOT_TYPE is (RESOURCE, ACTIVITY);

type PERIOD_ARRAY_TYPE is array(PERIOD_NUMBER_TYPE) of
                          TIMETABLE_NODE_PTR_TYPE;

type PERIOD_FACET_RECORD_TYPE(SLOT_KIND : PERIOD_SLOT_TYPE) is
record
  case SLOT_KIND is
    when RESOURCE => INITIALS     : SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
                     ROOM         : SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
                     WEEK_CODE    : SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
                     WEEK_ARRAY   : WEEK_ARRAY_TYPE;
    when ACTIVITY => NUMBER       : POSITIVE;
                     TOTAL        : POSITIVE;
```

```
                        ACTIVITY_PTR : TIMETABLE_NODE_PTR_TYPE;
  end case;
end record;
type PERIOD_FACET_RECORD_PTR_TYPE is access
   PERIOD_FACET_RECORD_TYPE;

function "<"
(LEFT_FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in PERIOD_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function ">"
(LEFT_FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in PERIOD_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function IS_EQUAL
(FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
 FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function IS_LESS_THAN
(FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
 FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function IS_GREATER_THAN
(FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
 FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE) return BOOLEAN;

procedure PUT
(FACET_PTR : in PERIOD_FACET_RECORD_PTR_TYPE);

procedure GET
(FACET_PTR : in out PERIOD_FACET_RECORD_PTR_TYPE;
 FROM_FILE : in out TEXT_IO.FILE_TYPE);

-------------------------------------------------------------------

package PERIOD_FRAME_BASE_PACKAGE is new
GENERIC_FRAME_BASE_PACKAGE
(SLOT_TYPE            => PERIOD_SLOT_TYPE,
 FACET_RECORD_TYPE    => PERIOD_FACET_RECORD_TYPE,
 FACET_RECORD_PTR_TYPE => PERIOD_FACET_RECORD_PTR_TYPE,
 "<"                  => "<",
 ">"                  => ">",
 IS_EQUAL             => IS_EQUAL,
 IS_LESS_THAN         => IS_LESS_THAN,
 IS_GREATER_THAN      => IS_GREATER_THAN,
 PUT                  => PUT,
 GET                  => GET);

-------------------------------------------------------------------

-- Staff are stored in a frame base under the three schools with
-- the following facet storing the details of each member of staff

type AVAILABILITY_MATRIX_TYPE is
     array(DAY_TYPE,
           PERIOD_NUMBER_TYPE) of WEEK_ARRAY_TYPE;

type STAFF_SLOT_TYPE is (DETAILS, SUBJECTS);
```

```
type STAFF_FACET_RECORD_TYPE(SLOT_KIND : STAFF_SLOT_TYPE) is
record
  case SLOT_KIND is
   when DETAILS   => SURNAME              : SYSTEM_TYPES_PACKAGE.
                                            DYNAMIC_STRING.
                                            STRING;
   when SUBJECTS  => MODULE               : SYSTEM_TYPES_PACKAGE.
                                            DYNAMIC_STRING.
                                            STRING;
                     AVAILABILITY_MATRIX :
                     AVAILABILITY_MATRIX_TYPE
                                            :=
                     (DAY_TYPE =>
                      (PERIOD_NUMBER_TYPE =>
                       (WEEK_NUMBER_TYPE => FALSE)));
  end case;

end record;

type STAFF_FACET_RECORD_PTR_TYPE is access
     STAFF_FACET_RECORD_TYPE;

function IS_EQUAL
(LEFT_FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE;
 RIGHT_FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function "<"
(LEFT_FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function ">"
(LEFT_FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function IS_EQUAL
(MODULE     : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function IS_LESS_THAN
(MODULE     : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN;

function IS_GREATER_THAN
(MODULE     : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN;

procedure PUT
(FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE);

procedure GET
(FACET_PTR : in out STAFF_FACET_RECORD_PTR_TYPE;
 FROM_FILE : in out TEXT_IO.FILE_TYPE);
```

```
-- Modules are stored in a frame base under the degree structure
-- with the following facet storing the details of each module

type MODULE_SLOT_TYPE is (STAFF);

type MODULE_FACET_RECORD_TYPE
     (SLOT_KIND : MODULE_SLOT_TYPE) is
record
  case SLOT_KIND is
    when STAFF => INITIALS : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
                  SURNAME  : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
  end case;

end record;

type MODULE_FACET_RECORD_PTR_TYPE is access
     MODULE_FACET_RECORD_TYPE;

function IS_EQUAL
(LEFT_FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE;
 RIGHT_FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function "<"
(LEFT_FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function ">"
(LEFT_FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function IS_EQUAL
(INITIALS    : in SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING.
                  STRING;
 FACET_PTR   : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function IS_LESS_THAN
(INITIALS    : in SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING.
                  STRING;
 FACET_PTR   : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN;

function IS_GREATER_THAN
(INITIALS    : in SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING.
                  STRING;
 FACET_PTR   : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN;
```

```
procedure PUT
(FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE);

procedure GET
(FACET_PTR : in out MODULE_FACET_RECORD_PTR_TYPE;
 FROM_FILE : in out TEXT_IO.FILE_TYPE);

-----------------------------------------------------------------

function IS_EQUAL
(TIMETABLE_NODE_PTR_1 : in TIMETABLE_NODE_PTR_TYPE;
 TIMETABLE_NODE_PTR_2 : in TIMETABLE_NODE_PTR_TYPE)
 return BOOLEAN;

-----------------------------------------------------------------

package TIMETABLE_LIST_PACKAGE
is new GENERIC_LIST_PACKAGE
       (TIMETABLE_NODE_PTR_TYPE,
        IS_EQUAL);

-----------------------------------------------------------------
function "<"
(TIMETABLE_NODE_PTR_1 : in TIMETABLE_NODE_PTR_TYPE ;
 TIMETABLE_NODE_PTR_2 : in TIMETABLE_NODE_PTR_TYPE
return BOOLEAN;

function ">"
(TIMETABLE_NODE_PTR_1 : in TIMETABLE_NODE_PTR_TYPE ;
 TIMETABLE_NODE_PTR_2 : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN;

function IS_EQUAL
(TIMETABLE_FACT_VALUE : in SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.STRING;
 TIMETABLE_NODE_PTR   : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN;

function IS_LESS_THAN
(TIMETABLE_FACT_VALUE : in SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.STRING;
 TIMETABLE_NODE_PTR   : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN;

function IS_GREATER_THAN
(TIMETABLE_FACT_VALUE : in SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.STRING;
 TIMETABLE_NODE_PTR   : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN;

procedure PUT
(TIMETABLE_NODE_PTR : in TIMETABLE_NODE_PTR_TYPE);

procedure PRINT
(LIST : in TIMETABLE_LIST_PACKAGE.LIST_TYPE);

-----------------------------------------------------------------

type TIMETABLE_NODE_RECORD
(KIND : TIMETABLE_NODE_KIND_TYPE := MODULE_ACTIVITY_KIND) is
```

```
record
     SUPPORTS    : TIMETABLE_LIST_PACKAGE.LIST_TYPE;
     ACTION      : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
     FACT        : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
     FUSED_FACT  : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
     RULE_PTR    : RULE_BASE_TYPES_PACKAGE.RULE_BASE_NODE_PTR_TYPE;
     SUPPORTERS  : TIMETABLE_LIST_PACKAGE.LIST_TYPE;
     STUDENTS    : STANDARD.NATURAL := 0;
   case KIND is
     when PERIOD_KIND =>
          PERIOD_DETAIL_FRAMES       : PERIOD_FRAME_BASE_PACKAGE.
                                       FRAME_BASE_RECORD_TYPE;
     when MODULE_ACTIVITY_KIND =>
          ACTIVITY                   : STANDARD.CHARACTER;
          LIST_OF_COMMON_ACTIVITIES  : SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING_LIST_PACKAGE.
                                       LIST_TYPE;
          STAFF_LIST                 : SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING_LIST_PACKAGE.
                                       LIST_TYPE;
          FREQUENCY                  : SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       STRING;
          FIRST_PERIOD               : PERIOD_NUMBER_TYPE;
          FIRST_PERIOD_STRING        : STANDARD.STRING(1..2);
          NUMBER_OF_PERIODS          : STANDARD.POSITIVE;
          HAS_PREFERENCE             : STANDARD.BOOLEAN;
          DAY_PREFERENCE             : SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       STRING;
          PERIOD_PREFERENCE          : SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       STRING;
     when MODULE_COMMON_REQUIREMENT_KIND =>
          LIST_OF_COMMON_MODULES     : SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING_LIST_PACKAGE.
                                       LIST_TYPE;
          NUMBER_OF_COMMON_MODULES   : STANDARD.POSITIVE := 1;
     when MODULE_REQUIREMENT_KIND =>
          null;
     when MODULE_KIND =>
          null;
   end case;
 end record;

-------------------------------------------------------------------

type BLACKBOARD_ITEM_TYPE is (LIST, FRAME);

type BLACKBOARD_ITEM_RECORD
     (ITEM_KIND : BLACKBOARD_ITEM_TYPE := LIST) is
record

  case ITEM_KIND is

    when LIST   =>  LIST   : TIMETABLE_LIST_PACKAGE.LIST_TYPE;
    when FRAME  =>  PERIOD : PERIOD_ARRAY_TYPE :=
```

```
                                    (PERIOD_NUMBER_TYPE =>
                                     new TIMETABLE_NODE_RECORD
                                         (PERIOD_KIND));
    end case;

  end record;

  type BLACKBOARD_ITEM_PTR_TYPE is access BLACKBOARD_ITEM_RECORD;

-------------------------------------------------------------------------
end TIMETABLE_TYPES_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit        : TIMETABLE_TYPES_PACKAGE body
-- Author      : A Harrison Software Engineering Group,
--               Cranfield University, RMCS, Shrivenham
-- Date        : 21 December 1991
-- Function    : This package provides the TIMETABLE application
--               definitions
--
-------------------------------------------------------------------
package body TIMETABLE_TYPES_PACKAGE is
-------------------------------------------------------------------

  -- Operations on period facets
  -----------------------------------------------------------------
  function "<"
  -----------------------------------------------------------------
  (LEFT_FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE ;
   RIGHT_FACET_PTR : in PERIOD_FACET_RECORD_PTR_TYPE)
   return BOOLEAN is
  -----------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_LESS_THAN(LEFT  => LEFT_FACET_PTR.
                                 INITIALS,
                        RIGHT => RIGHT_FACET_PTR.
                                 INITIALS);
  -----------------------------------------------------------------
  end "<";
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  function ">"
  -----------------------------------------------------------------
  (LEFT_FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE ;
   RIGHT_FACET_PTR : in PERIOD_FACET_RECORD_PTR_TYPE)
   return BOOLEAN is
  -----------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_GREATER_THAN(LEFT  => LEFT_FACET_PTR.
                                    INITIALS,
                           RIGHT => RIGHT_FACET_PTR.
                                    INITIALS);
  -----------------------------------------------------------------
  end ">";
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  function IS_EQUAL
  -----------------------------------------------------------------
  (FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE) return BOOLEAN is
  -----------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
```

```
          DYNAMIC_STRING.
          IS_EQUAL(LEFT  => FACET_NAME,
                   RIGHT => FACET_PTR.
                            INITIALS);
  ------------------------------------------------------------------
  end IS_EQUAL;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function IS_LESS_THAN
  ------------------------------------------------------------------
  (FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_LESS_THAN(LEFT  => FACET_NAME,
                        RIGHT => FACET_PTR.
                                 INITIALS);
  ------------------------------------------------------------------
  end IS_LESS_THAN;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function IS_GREATER_THAN
  ------------------------------------------------------------------
  (FACET_NAME : in SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
   FACET_PTR  : in PERIOD_FACET_RECORD_PTR_TYPE) return BOOLEAN is
  ------------------------------------------------------------------
  begin
    return SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           IS_GREATER_THAN(LEFT  => FACET_NAME,
                           RIGHT => FACET_PTR.
                                    INITIALS);
  ------------------------------------------------------------------
  end IS_GREATER_THAN;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure PUT
  ------------------------------------------------------------------
  (FACET_PTR : in PERIOD_FACET_RECORD_PTR_TYPE) is
  ------------------------------------------------------------------
  begin
    TEXT_IO.SET_COL(15);
    case FACET_PTR.SLOT_KIND is
      when RESOURCE => SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING.
                       PUT
                       (THE_STRING => FACET_PTR.INITIALS);
                        TEXT_IO.PUT(' ');
                        SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        PUT
                        (THE_STRING => FACET_PTR.WEEK_CODE);
```

```
                        TEXT_IO.NEW_LINE;
    when ACTIVITY => INTEGER_TEXT_IO.
                     PUT
                     (FACET_PTR.NUMBER, 1);
                     TEXT_IO.PUT(" of ");
                     INTEGER_TEXT_IO.
                     PUT
                     (FACET_PTR.TOTAL, 1);
                     TEXT_IO.NEW_LINE;
    end case;
-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(FACET_PTR : in out PERIOD_FACET_RECORD_PTR_TYPE;
 FROM_FILE : in out TEXT_IO.FILE_TYPE) is
-------------------------------------------------------------------
begin
  null; -- Dummy
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------

-- Operations on Staff facets
-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(LEFT_FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE;
 RIGHT_FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE)
 return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL(LEFT  => LEFT_FACET_PTR.MODULE,
                  RIGHT => RIGHT_FACET_PTR.MODULE);
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function "<"
-------------------------------------------------------------------
(LEFT_FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_LESS_THAN(LEFT  => LEFT_FACET_PTR.MODULE,
                      RIGHT => RIGHT_FACET_PTR.MODULE);
-------------------------------------------------------------------
end "<";
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
function ">"
-------------------------------------------------------------------
(LEFT_FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_GREATER_THAN(LEFT  => LEFT_FACET_PTR.MODULE,
                         RIGHT => RIGHT_FACET_PTR.MODULE);
-------------------------------------------------------------------
end ">";
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(MODULE     : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL(LEFT  => MODULE,
                  RIGHT => FACET_PTR.MODULE);
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_LESS_THAN
(MODULE     : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_LESS_THAN(LEFT  => MODULE,
                      RIGHT => FACET_PTR.MODULE);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_GREATER_THAN
-------------------------------------------------------------------
(MODULE     : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
```

```
 FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE) return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_GREATER_THAN(LEFT  => MODULE,
                         RIGHT => FACET_PTR.MODULE);
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PUT
-------------------------------------------------------------------
(FACET_PTR : in STAFF_FACET_RECORD_PTR_TYPE) is
-------------------------------------------------------------------
begin

  TEXT_IO.SET_COL(15);

  case FACET_PTR.SLOT_KIND is

    when DETAILS  => SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     PUT
                     (THE_STRING => FACET_PTR.SURNAME);

    when SUBJECTS => SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     PUT
                     (THE_STRING => FACET_PTR.MODULE);
  end case;

  TEXT_IO.NEW_LINE;

-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(FACET_PTR : in out STAFF_FACET_RECORD_PTR_TYPE;
 FROM_FILE : in out TEXT_IO.FILE_TYPE) is

  MODULE : STANDARD.STRING(1..80);

  LAST : POSITIVE range 1 .. 80;
-------------------------------------------------------------------
begin

  -- Get facet

  TEXT_IO.
  GET_LINE
  (FILE => FROM_FILE,
   ITEM => MODULE,
   LAST => LAST);
```

```
    -- Store facet

   case FACET_PTR.SLOT_KIND is

     when DETAILS  => SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      COPY
                      (FROM_THE_SUBSTRING => MODULE
                                             (1..LAST),
                       TO_THE_STRING      => FACET_PTR.SURNAME);
     when SUBJECTS => SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      COPY
                     (FROM_THE_SUBSTRING => MODULE
                                            (1..LAST),
                   .. TO_THE_STRING      => FACET_PTR.MODULE);
   end case;
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------

-- Operations on Module facets

-------------------------------------------------------------------
function IS_EQUAL
-------------------------------------------------------------------
(LEFT_FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE;
 RIGHT_FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin
   return SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
          IS_EQUAL(LEFT  => LEFT_FACET_PTR.INITIALS,
                   RIGHT => RIGHT_FACET_PTR.INITIALS);
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
function "<"
-------------------------------------------------------------------
(LEFT_FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin
   return SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
          IS_LESS_THAN(LEFT  => LEFT_FACET_PTR.INITIALS,
                       RIGHT => RIGHT_FACET_PTR.INITIALS);
-------------------------------------------------------------------
end "<";
-------------------------------------------------------------------


-------------------------------------------------------------------
function ">"
```

```
--------------------------------------------------------------------
(LEFT_FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE ;
 RIGHT_FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE)
return BOOLEAN is
--------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_GREATER_THAN(LEFT  => LEFT_FACET_PTR.INITIALS,
                         RIGHT => RIGHT_FACET_PTR.INITIALS);
--------------------------------------------------------------------
end ">";
--------------------------------------------------------------------


--------------------------------------------------------------------
function IS_EQUAL
--------------------------------------------------------------------
(INITIALS   : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE) return BOOLEAN is
--------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_EQUAL(LEFT  => INITIALS,
                  RIGHT => FACET_PTR.INITIALS);
--------------------------------------------------------------------
end IS_EQUAL;
--------------------------------------------------------------------


--------------------------------------------------------------------
function IS_LESS_THAN
(INITIALS   : in SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;
 FACET_PTR  : in MODULE_FACET_RECORD_PTR_TYPE) return BOOLEAN is
--------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         IS_LESS_THAN(LEFT  => INITIALS,
                      RIGHT => FACET_PTR.INITIALS);
--------------------------------------------------------------------
end IS_LESS_THAN;
--------------------------------------------------------------------


--------------------------------------------------------------------
function IS_GREATER_THAN
--------------------------------------------------------------------
(INITIALS  : in SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
 FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE) return BOOLEAN is
--------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
```

```
          IS_GREATER_THAN(LEFT  => INITIALS,
                          RIGHT => FACET_PTR.INITIALS);
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PUT
-------------------------------------------------------------------
(FACET_PTR : in MODULE_FACET_RECORD_PTR_TYPE) is
-------------------------------------------------------------------
begin

  TEXT_IO.SET_COL(15);

  case FACET_PTR.SLOT_KIND is
    when STAFF  => SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   PUT
                   (THE_STRING => FACET_PTR.INITIALS);

                    TEXT_IO.NEW_LINE;
                    TEXT_IO.SET_COL(15);

                    SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    PUT
                   (THE_STRING => FACET_PTR.SURNAME);

  end case;

  TEXT_IO.NEW_LINE;

-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(FACET_PTR : in out MODULE_FACET_RECORD_PTR_TYPE;
 FROM_FILE : in out TEXT_IO.FILE_TYPE) is

  INITIALS : STANDARD.STRING(1 .. 80);
  SURNAME  : STANDARD.STRING(1 .. 80);

  LAST_INITIAL : POSITIVE range 1 .. 80;
  LAST_SURNAME : POSITIVE range 1 .. 80;
-------------------------------------------------------------------
begin

  -- Get initials

  TEXT_IO.
  GET_LINE
  (FILE => FROM_FILE,
   ITEM => INITIALS,
   LAST => LAST_INITIAL);
```

```
   -- Get surname

   TEXT_IO.
   GET_LINE
   (FILE => FROM_FILE,
    ITEM => SURNAME,
    LAST => LAST_SURNAME);

   -- Store in facet

   case FACET_PTR.SLOT_KIND is

     when STAFF  => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    COPY
                    (FROM_THE_SUBSTRING => INITIALS
                                           (1..LAST_INITIAL),
                     TO_THE_STRING      => FACET_PTR.INITIALS);

                    SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    COPY
                    (FROM_THE_SUBSTRING => SURNAME
                                           (1..LAST_SURNAME);
                     TO_THE_STRING      => FACET_PTR.SURNAME);

   end case;
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------

-- Operations on nodes

-------------------------------------------------------------------
--
-- Is_equal checks for the equality of two TIMETABLE facts
--
-------------------------------------------------------------------
function IS_EQUAL
(TIMETABLE_NODE_PTR_1 : in TIMETABLE_NODE_PTR_TYPE;
 TIMETABLE_NODE_PTR_2 : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin

   return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.
          IS_EQUAL(TIMETABLE_NODE_PTR_1.FACT,
                   TIMETABLE_NODE_PTR_2.FACT);
-------------------------------------------------------------------
end IS_EQUAL;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- "<" checks two TIMETABLE facts
--
-------------------------------------------------------------------
function "<"
```

```
------------------------------------------------------------------
(TIMETABLE_NODE_PTR_1 : in TIMETABLE_NODE_PTR_TYPE;
 TIMETABLE_NODE_PTR_2 : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
------------------------------------------------------------------
begin

   return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.
          IS_LESS_THAN(TIMETABLE_NODE_PTR_1.FACT,
                       TIMETABLE_NODE_PTR_2.FACT);
------------------------------------------------------------------
end "<";
------------------------------------------------------------------


------------------------------------------------------------------
--
-- ">" checks two TIMETABLE facts
--
------------------------------------------------------------------
function ">"
------------------------------------------------------------------
(TIMETABLE_NODE_PTR_1 : in TIMETABLE_NODE_PTR_TYPE;
 TIMETABLE_NODE_PTR_2 : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
------------------------------------------------------------------
begin

  return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.
         IS_GREATER_THAN(TIMETABLE_NODE_PTR_1.FACT,
                         TIMETABLE_NODE_PTR_2.FACT);
------------------------------------------------------------------
end ">";
------------------------------------------------------------------


------------------------------------------------------------------
--
-- Is_equal checks two facts given a fact value and a pointer to a
-- fact
--
------------------------------------------------------------------
function IS_EQUAL
------------------------------------------------------------------
(TIMETABLE_FACT_VALUE : in SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.STRING;
 TIMETABLE_NODE_PTR   : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.
         IS_EQUAL(TIMETABLE_FACT_VALUE,
                  TIMETABLE_NODE_PTR.FACT);
------------------------------------------------------------------
end IS_EQUAL;
------------------------------------------------------------------


------------------------------------------------------------------
--
```

```
-- Is_less_than checks two facts given a fact value and a pointer
-- to a fact
--
-------------------------------------------------------------------
function IS_LESS_THAN
-------------------------------------------------------------------
(TIMETABLE_FACT_VALUE : in SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.STRING;
 TIMETABLE_NODE_PTR   : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin
  return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.
         IS_LESS_THAN(TIMETABLE_FACT_VALUE,
                      TIMETABLE_NODE_PTR.FACT);
-------------------------------------------------------------------
end IS_LESS_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Is_greater_than checks two facts given a fact value and a
-- pointer to a fact
--
-------------------------------------------------------------------
function IS_GREATER_THAN
-------------------------------------------------------------------
(TIMETABLE_FACT_VALUE : in SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.STRING;
 TIMETABLE_NODE_PTR   : in TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
begin

  return SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.
         IS_GREATER_THAN(TIMETABLE_FACT_VALUE,
                         TIMETABLE_NODE_PTR.FACT);
-------------------------------------------------------------------
end IS_GREATER_THAN;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Put a fact to the VDU
--
-------------------------------------------------------------------
procedure PUT
-------------------------------------------------------------------
(TIMETABLE_NODE_PTR : in TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
begin

  case TIMETABLE_NODE_PTR.KIND is

    when PERIOD_KIND => null;

    when MODULE_ACTIVITY_KIND => SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING.
```

```
                                        PUT(TIMETABLE_NODE_PTR.
                                            FUSED_FACT);

    when MODULE_COMMON_REQUIREMENT_KIND => SYSTEM_TYPES_PACKAGE.
                                           DYNAMIC_STRING.
                                           PUT(TIMETABLE_NODE_PTR.
                                               FUSED_FACT);

    when MODULE_REQUIREMENT_KIND => SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    PUT
                                    (TIMETABLE_NODE_PTR.
                                     FUSED_FACT);

    when MODULE_KIND => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        PUT(TIMETABLE_NODE_PTR.FUSED_FACT);

  end case;
-------------------------------------------------------------------
end PUT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Print the contents of a list
--
-------------------------------------------------------------------
procedure PRINT
-------------------------------------------------------------------
(LIST : in TIMETABLE_LIST_PACKAGE.LIST_TYPE) is

  COLUMN : TEXT_IO.POSITIVE_COUNT := 1;

begin

  for INDEX in 1 .. TIMETABLE_LIST_PACKAGE.LENGTH_OF(LIST)
  loop

    if NATURAL(TEXT_IO.COL) +
       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       LENGTH_OF
       (THE_STRING => TIMETABLE_LIST_PACKAGE.
        ITEM_AT
        (LIST        => LIST,
         NODE_NUMBER => INDEX).FUSED_FACT) > 80 then

      TEXT_IO.NEW_LINE;
      COLUMN := 1;

    end if;

    PUT(TIMETABLE_LIST_PACKAGE.ITEM_AT(LIST, INDEX));

    case TIMETABLE_LIST_PACKAGE.
         ITEM_AT(LIST, INDEX).KIND is

      when PERIOD_KIND |
```

```
            MODULE_ACTIVITY_KIND|
            MODULE_COMMON_REQUIREMENT_KIND => COLUMN :=
                                              COLUMN + 40;

        when others => COLUMN := COLUMN + 20;

      end case;

      TEXT_IO.SET_COL(COLUMN);

    end loop;

    TEXT_IO.NEW_LINE;
-------------------------------------------------------------------------
  end PRINT;
-------------------------------------------------------------------------

-------------------------------------------------------------------------
end TIMETABLE_TYPES_PACKAGE;
-------------------------------------------------------------------------
```

Timetable Blackboard

```
-------------------------------------------------------------------------
--
-- Unit      : TIME_TABLE_BLACKBOARD_PACKAGE
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 2 January 1992
-- Function  : An instantiation of a blackboard to meet the
--             application requirement
--
-------------------------------------------------------------------------
with TIMETABLE_TYPES_PACKAGE,
     GENERIC_BLACKBOARD_PACKAGE;
-------------------------------------------------------------------------
package TIMETABLE_BLACKBOARD_PACKAGE is
-------------------------------------------------------------------------

  package TIMETABLE_BLACKBOARD is new
  GENERIC_BLACKBOARD_PACKAGE
  (ITEM_TYPE        => TIMETABLE_TYPES_PACKAGE.
                       BLACKBOARD_ITEM_RECORD,
   ITEM_PTR_TYPE    => TIMETABLE_TYPES_PACKAGE.
                       BLACKBOARD_ITEM_PTR_TYPE,
   LEVEL_INDEX_TYPE => TIMETABLE_TYPES_PACKAGE.TIMETABLE_LEVEL_TYPE,
   ITEM_INDEX_TYPE  => TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE);

  -----------------------------------------------------------------------

  BLACKBOARD : TIMETABLE_BLACKBOARD.BLACKBOARD_TYPE;

  procedure SWAP_TIMETABLE_ITEM
  (FROM_DAY    : in TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
   FROM_PERIOD : in TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
   TO_DAY      : in TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
   TO_PERIOD   : in TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE);
-------------------------------------------------------------------------
end TIMETABLE_BLACKBOARD_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : TIME_TABLE_BLACKBOARD_PACKAGE
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 6 October 1993
-- Function  : An instantiation of a blackboard to meet the
--             application requirement
--
-------------------------------------------------------------------
with TIMETABLE_TYPES_PACKAGE,
     GENERIC_BLACKBOARD_PACKAGE;
-------------------------------------------------------------------
package body TIMETABLE_BLACKBOARD_PACKAGE is
-------------------------------------------------------------------

  -----------------------------------------------------------------
  procedure SWAP_TIMETABLE_ITEM
  -----------------------------------------------------------------
  (FROM_DAY    : in TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
   FROM_PERIOD : in TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
   TO_DAY      : in TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
   TO_PERIOD   : in TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE) is

    TEMP : TIMETABLE_TYPES_PACKAGE.TIMETABLE_NODE_PTR_TYPE;
  -----------------------------------------------------------------
  begin

    TEMP := BLACKBOARD(TIMETABLE_TYPES_PACKAGE.DAYS)
                      (FROM_DAY).PERIOD(FROM_PERIOD);
    BLACKBOARD(TIMETABLE_TYPES_PACKAGE.DAYS)
              (FROM_DAY).PERIOD(FROM_PERIOD):=
    BLACKBOARD(TIMETABLE_TYPES_PACKAGE.DAYS)
              (TO_DAY).PERIOD(TO_PERIOD);
    BLACKBOARD(TIMETABLE_TYPES_PACKAGE.DAYS)
              (TO_DAY).PERIOD(TO_PERIOD):= TEMP;

  -----------------------------------------------------------------
  end SWAP_TIMETABLE_ITEM;
  -----------------------------------------------------------------

-------------------------------------------------------------------
end TIMETABLE_BLACKBOARD_PACKAGE;
-------------------------------------------------------------------
```

Blackboard Initialise KS

```
------------------------------------------------------------------------
--
-- Unit      : BLACKBOARD_INITIALISE_KS_PACKAGE specification
-- Author    : A Harrison Software Enginering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 2 January 1992
-- Function  : This package sets each component of the blackboard to
--             a list or a tree as required by the application
--
------------------------------------------------------------------------
with TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
package BLACKBOARD_INITIALISE_KS_PACKAGE is
------------------------------------------------------------------------

  procedure BUILD_BLACKBOARD_LEVELS
  (BLACKBOARD : in out TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_TYPE);

------------------------------------------------------------------------
end BLACKBOARD_INITIALISE_KS_PACKAGE;
------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unit      : BLACKBOARD_INITIALISE_KS_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 2 January 1992
-- Function  : This package sets each component of the blackboard as
--             required by the application
--
------------------------------------------------------------------

------------------------------------------------------------------
package body BLACKBOARD_INITIALISE_KS_PACKAGE is
------------------------------------------------------------------

  ----------------------------------------------------------------
  --
  -- Build_blackboard_levels builds the blackboard to the
  --  application requirement
  --
  ----------------------------------------------------------------
  procedure BUILD_BLACKBOARD_LEVELS
  ----------------------------------------------------------------
  (BLACKBOARD : in out TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_TYPE) is
    ----------------------------------------------------------------

    ----------------------------------------------------------------
    --
    -- Initialise_level sets the length of a single level to the
    -- user requirement
    --
    ----------------------------------------------------------------
    procedure INITIALISE_LEVEL
    ----------------------------------------------------------------
    (ITEM_TYPE    : in     TIMETABLE_TYPES_PACKAGE.
                           BLACKBOARD_ITEM_TYPE :=
                           TIMETABLE_TYPES_PACKAGE.
                           LIST;
     BLACKBOARD   : in out TIMETABLE_BLACKBOARD_PACKAGE.
                           TIMETABLE_BLACKBOARD.BLACKBOARD_TYPE;
     LEVEL_INDEX  : in     TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_LEVEL_TYPE;
     FROM_INDEX   : in     TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_ITEM_TYPE;
     TO_INDEX     : in     TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_ITEM_TYPE) is
      ----------------------------------------------------------------
      NEW_ITEM : TIMETABLE_TYPES_PACKAGE.BLACKBOARD_ITEM_PTR_TYPE;
    ----------------------------------------------------------------
    begin -- Initialise_Level

      for ITEM_INDEX in FROM_INDEX .. TO_INDEX
      loop

        NEW_ITEM := new TIMETABLE_TYPES_PACKAGE.
                        BLACKBOARD_ITEM_RECORD(ITEM_TYPE);

        TIMETABLE_BLACKBOARD_PACKAGE.
```

```
      TIMETABLE_BLACKBOARD.
      PUT_BLACKBOARD_ITEM(BLACKBOARD  => BLACKBOARD,
                          ITEM        => NEW_ITEM,
                          LEVEL_INDEX => LEVEL_INDEX,
                          ITEM_INDEX  => ITEM_INDEX);

    end loop;
  ------------------------------------------------------------------
  end INITIALISE_LEVEL;
  ------------------------------------------------------------------
begin -- Build_Blackboard_Levels

  --  Construct day level

  TIMETABLE_BLACKBOARD_PACKAGE.
  TIMETABLE_BLACKBOARD.
  CONSTRUCT_BLACKBOARD
  (BLACKBOARD => BLACKBOARD,
   LEVEL      => TIMETABLE_TYPES_PACKAGE.DAYS,
   FROM       => TIMETABLE_TYPES_PACKAGE.MONDAY,
   TO         => TIMETABLE_TYPES_PACKAGE.FRIDAY);

  INITIALISE_LEVEL
  (ITEM_TYPE   => TIMETABLE_TYPES_PACKAGE.
                  FRAME,
   BLACKBOARD  => BLACKBOARD,
   LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.DAYS,
   FROM_INDEX  => TIMETABLE_TYPES_PACKAGE.MONDAY,
   TO_INDEX    => TIMETABLE_TYPES_PACKAGE.FRIDAY);

  -- Initialise each period frame base

  for DAY in TIMETABLE_TYPES_PACKAGE.
             TIMETABLE_ITEM_TYPE'(TIMETABLE_TYPES_PACKAGE.MONDAY)
             TIMETABLE_TYPES_PACKAGE.
             TIMETABLE_ITEM_TYPE'(TIMETABLE_TYPES_PACKAGE.FRIDAY)
  loop

    for PERIOD_NUMBER in TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE
    loop

      TIMETABLE_TYPES_PACKAGE.
      PERIOD_FRAME_BASE_PACKAGE.
      INITIALISE_FRAME_BASE
      (FRAME_BASE_RECORD =>
       TIMETABLE_BLACKBOARD_PACKAGE.
       TIMETABLE_BLACKBOARD.
       BLACKBOARD_ITEM
       (BLACKBOARD  => BLACKBOARD,
        LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                       DAYS,
        ITEM_INDEX  => DAY).PERIOD(PERIOD_NUMBER).
                               PERIOD_DETAIL_FRAMES);
    end loop;

  end loop;


  -- Construct degree activities level
```

```
TIMETABLE_BLACKBOARD_PACKAGE.
TIMETABLE_BLACKBOARD.
CONSTRUCT_BLACKBOARD
(BLACKBOARD => BLACKBOARD,
 LEVEL      => TIMETABLE_TYPES_PACKAGE.DEGREE_ACTIVITIES,
 FROM       => TIMETABLE_TYPES_PACKAGE.LECTURES,
 TO         => TIMETABLE_TYPES_PACKAGE.PRACTICALS);

INITIALISE_LEVEL
(BLACKBOARD  => BLACKBOARD,
 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.DEGREE_ACTIVITIES,
 FROM_INDEX  => TIMETABLE_TYPES_PACKAGE.LECTURES,
 TO_INDEX    => TIMETABLE_TYPES_PACKAGE.PRACTICALS);

-- Construct common modules level

TIMETABLE_BLACKBOARD_PACKAGE.
TIMETABLE_BLACKBOARD.
CONSTRUCT_BLACKBOARD
(BLACKBOARD => BLACKBOARD,
 LEVEL      => TIMETABLE_TYPES_PACKAGE.
               COMMON_MODULE_REQUIREMENTS,
 FROM       => TIMETABLE_TYPES_PACKAGE.
               COMMON_MODULE_REQUIREMENT_LIST,
 TO         => TIMETABLE_TYPES_PACKAGE.
               COMMON_MODULE_REQUIREMENT_LIST);

INITIALISE_LEVEL
(BLACKBOARD  => BLACKBOARD,
 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                COMMON_MODULE_REQUIREMENTS,
 FROM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                COMMON_MODULE_REQUIREMENT_LIST,
 TO_INDEX    => TIMETABLE_TYPES_PACKAGE.
                COMMON_MODULE_REQUIREMENT_LIST);

-- Construct module requirement level

TIMETABLE_BLACKBOARD_PACKAGE.
TIMETABLE_BLACKBOARD.
CONSTRUCT_BLACKBOARD
(BLACKBOARD => BLACKBOARD,
 LEVEL      => TIMETABLE_TYPES_PACKAGE.MODULE_REQUIREMENTS,
 FROM       => TIMETABLE_TYPES_PACKAGE.ESE,
 TO         => TIMETABLE_TYPES_PACKAGE.IT);

INITIALISE_LEVEL
(BLACKBOARD  => BLACKBOARD,
 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.MODULE_REQUIREMENTS,
 FROM_INDEX  => TIMETABLE_TYPES_PACKAGE.ESE,
 TO_INDEX    => TIMETABLE_TYPES_PACKAGE.IT);

-- Construct degree modules level

TIMETABLE_BLACKBOARD_PACKAGE.
TIMETABLE_BLACKBOARD.
CONSTRUCT_BLACKBOARD
(BLACKBOARD => BLACKBOARD,
 LEVEL      => TIMETABLE_TYPES_PACKAGE.DEGREE_MODULES,
```

```
      FROM        => TIMETABLE_TYPES_PACKAGE.ESE,
      TO          => TIMETABLE_TYPES_PACKAGE.IT);

    INITIALISE_LEVEL
    (BLACKBOARD  => BLACKBOARD,
     LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.DEGREE_MODULES,
     FROM_INDEX  => TIMETABLE_TYPES_PACKAGE.ESE,
     TO_INDEX    => TIMETABLE_TYPES_PACKAGE.IT);

    -- Construct event lists level

    TIMETABLE_BLACKBOARD_PACKAGE.
    TIMETABLE_BLACKBOARD.
    CONSTRUCT_BLACKBOARD
    (BLACKBOARD => BLACKBOARD,
     LEVEL      => TIMETABLE_TYPES_PACKAGE.EVENT_LISTS,
     FROM       => TIMETABLE_TYPES_PACKAGE.COMMON_EVENTS,
     TO         => TIMETABLE_TYPES_PACKAGE.PERIOD_EVENTS);

    INITIALISE_LEVEL
    (BLACKBOARD  => BLACKBOARD,
     LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.EVENT_LISTS,
     FROM_INDEX  => TIMETABLE_TYPES_PACKAGE.COMMON_EVENTS,
     TO_INDEX    => TIMETABLE_TYPES_PACKAGE.PERIOD_EVENTS);

  ------------------------------------------------------------------
  end BUILD_BLACKBOARD_LEVELS;
  ------------------------------------------------------------------

------------------------------------------------------------------
end BLACKBOARD_INITIALISE_KS_PACKAGE;
------------------------------------------------------------------
```

Syllabus KS

```
------------------------------------------------------------------------
--
-- Unit      : SYLLABUS_KS_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 29 December 1991
-- Function  : This package provides the operation that gets the
--             degree module codes from a text and places them on the
--             event list.
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     RULE_BASE_TYPES_PACKAGE,
     GENERIC_RULE_BASE_INFERENCE_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
package SYLLABUS_KS_PACKAGE is
------------------------------------------------------------------------

  procedure GET_DEGREE_MODULES
   (FROM_FILENAME : in     STANDARD.STRING;
    BLACKBOARD    : in out TIMETABLE_BLACKBOARD_PACKAGE.
                           TIMETABLE_BLACKBOARD.
                           BLACKBOARD_TYPE);

------------------------------------------------------------------------
end SYLLABUS_KS_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Title     : SYLLABUS_KS_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 29 December 1991
-- Function  : This package provides the operation that gets the
--             degree module codes from a text file and places them
--             on the blackboard event list.
--
-------------------------------------------------------------------
package body SYLLABUS_KS_PACKAGE is
-------------------------------------------------------------------

   -- Integrate a rule abstract knowledge type

   package RULE_BASE_PACKAGE is new
   GENERIC_RULE_BASE_INFERENCE_PACKAGE
   (RULE_BASES_FILENAME => "plmsubfile.list",
    FACT_COMPOSITE_TYPE => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_RECORD,
    FACT_PTR_TYPE       => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE,
    "<"                 => TIMETABLE_TYPES_PACKAGE."<",
    ">"                 => TIMETABLE_TYPES_PACKAGE.">",
    IS_EQUAL            => TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
    IS_LESS_THAN        => TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
    IS_GREATER_THAN     => TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
    PUT                 => TIMETABLE_TYPES_PACKAGE.PUT);


  -------------------------------------------------------------------
  --
  -- Put_on_degree_modules_level adds the current event module to
  -- the appropriate blackboard list
  --
  -------------------------------------------------------------------
  procedure PUT_ON_DEGREE_MODULES_LEVEL
  -------------------------------------------------------------------
  (BLACKBOARD  : in out TIMETABLE_BLACKBOARD_PACKAGE.
                        TIMETABLE_BLACKBOARD.
                        BLACKBOARD_TYPE;
   MODULE_PTR : in      TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE;
   MODULE     : in      STRING) is
  -------------------------------------------------------------------
  begin

    if MODULE(1) = 'E'then

      TIMETABLE_TYPES_PACKAGE.
      TIMETABLE_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST        => TIMETABLE_BLACKBOARD_PACKAGE.
                      TIMETABLE_BLACKBOARD.
                      BLACKBOARD_ITEM
                      (BLACKBOARD  => BLACKBOARD,
                       LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                      DEGREE_MODULES,
                       ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
```

```
                                        ESE).LIST,
      THIS_ITEM => MODULE_PTR);
  elsif MODULE(1) ='I' then

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DEGREE_MODULES,
                    ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.IT).
                                   LIST,
     THIS_ITEM => MODULE_PTR);

  end if;

-------------------------------------------------------------------
end PUT_ON_DEGREE_MODULES_LEVEL;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Fire rules on agenda
--
-------------------------------------------------------------------
procedure FIRE_RULES
-------------------------------------------------------------------
(AGENDA              : in out RULE_BASE_TYPES_PACKAGE.
                              AGENDA_LIST_PACKAGE.
                              LIST_TYPE;
 MODULE_PTR          : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_NODE_PTR_TYPE;
 NUMBER_OF_STUDENTS  : in     STANDARD.NATURAL;
 BLACKBOARD          : in out TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_TYPE). is
-------------------------------------------------------------------

  RULE_PTR : RULE_BASE_TYPES_PACKAGE.
             RULE_BASE_NODE_PTR_TYPE;

  NEW_MODULE_PTR : TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;

-------------------------------------------------------------------
begin

  -- get first rule for submodule to replace parent module

    RULE_BASE_TYPES_PACKAGE.
    AGENDA_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST       => AGENDA,
     THIS_ITEM => RULE_PTR);

 -- Copy this to parent node
```

```
SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_STRING =>RULE_PTR.ACTION.ACTION_TO_DO,
 TO_THE_STRING   => MODULE_PTR.FACT);
SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_STRING =>RULE_PTR.ACTION.ACTION_TO_DO,
 TO_THE_STRING   => MODULE_PTR.FUSED_FACT);

     -- Put on event list

 TIMETABLE_TYPES_PACKAGE.
 TIMETABLE_LIST_PACKAGE.
 PUT_ON_BACK_OF
 (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                TIMETABLE_BLACKBOARD.
                BLACKBOARD_ITEM
                (BLACKBOARD  => BLACKBOARD,
                 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                EVENT_LISTS,
                 ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                COMMON_EVENTS).LIST,
  THIS_ITEM => MODULE_PTR);

  -- Put on degree_modules level of blackboard

  PUT_ON_DEGREE_MODULES_LEVEL
  (BLACKBOARD => BLACKBOARD,
   MODULE_PTR => MODULE_PTR,
   MODULE     => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => MODULE_PTR.FACT));

-- Add rest of submodules

while not RULE_BASE_TYPES_PACKAGE.
          AGENDA_LIST_PACKAGE.
          IS_EMPTY
          (LIST => AGENDA)
loop

  -- Get the rule

  RULE_BASE_TYPES_PACKAGE.
  AGENDA_LIST_PACKAGE.
  GET_FROM_FRONT_OF
  (LIST      => AGENDA,
   THIS_ITEM => RULE_PTR);

  -- Create new submodule

  NEW_MODULE_PTR := new TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_RECORD
                        (TIMETABLE_TYPES_PACKAGE.MODULE_KIND);

  NEW_MODULE_PTR.STUDENTS := NUMBER_OF_STUDENTS;
```

```
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_STRING => RULE_PTR.ACTION.ACTION_TO_DO,
    TO_THE_STRING   => NEW_MODULE_PTR.FACT);

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_STRING => RULE_PTR.ACTION.ACTION_TO_DO,
    TO_THE_STRING   => NEW_MODULE_PTR.FUSED_FACT);

    -- Put on event list

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   EVENT_LISTS,
                    ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                   COMMON_EVENTS).LIST,
     THIS_ITEM => NEW_MODULE_PTR);

     -- Put on degree_modules level of blackboard

     PUT_ON_DEGREE_MODULES_LEVEL
     (BLACKBOARD => BLACKBOARD,
      MODULE_PTR => NEW_MODULE_PTR,
      MODULE     => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => MODULE_PTR.FACT));

  end loop;

-------------------------------------------------------------------
end FIRE_RULES;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Create the node for the new module
--
-------------------------------------------------------------------
procedure CREATE(MODULE     : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     STRING;
                 STUDENTS   : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     STRING;
                 BLACKBOARD : in out TIMETABLE_BLACKBOARD_PACKAGE.
                                     TIMETABLE_BLACKBOARD.
                                     BLACKBOARD_TYPE) is
-------------------------------------------------------------------
```

```
   MODULE_PTR : TIMETABLE_TYPES_PACKAGE.
                TIMETABLE_NODE_PTR_TYPE;

   AGENDA     : RULE_BASE_TYPES_PACKAGE.
                AGENDA_LIST_PACKAGE.
                LIST_TYPE;

   NUMBER_OF_STUDENTS : STANDARD.NATURAL;
-------------------------------------------------------------------
begin

   MODULE_PTR := new TIMETABLE_TYPES_PACKAGE.TIMETABLE_NODE_RECORD
                     (TIMETABLE_TYPES_PACKAGE.MODULE_KIND);

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_STRING    => MODULE,
    TO_THE_STRING => MODULE_PTR.FACT);

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_STRING => MODULE,
    TO_THE_STRING => MODULE_PTR.FUSED_FACT);

   NUMBER_OF_STUDENTS := STANDARD.NATURAL'
                         VALUE
                         (SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          SUBSTRING_OF
                          (THE_STRING => STUDENTS));

   MODULE_PTR.
   STUDENTS := NUMBER_OF_STUDENTS;

   RULE_BASE_PACKAGE.
   INFERENCE
   (RULE_BASE => SYSTEM_TYPES_PACKAGE.
                 GET_FIELD
                 (NUMBER => 1,
                  FROM   => SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            SUBSTRING_OF
                            (THE_STRING => MODULE_PTR.FACT)),
    FACT_PTR  => MODULE_PTR,
    AGENDA    => AGENDA);

   if not RULE_BASE_TYPES_PACKAGE.
          AGENDA_LIST_PACKAGE.
          IS_EMPTY
          (AGENDA) then

     FIRE_RULES
     (AGENDA              => AGENDA,
      MODULE_PTR          => MODULE_PTR,
      NUMBER_OF_STUDENTS  => NUMBER_OF_STUDENTS,
      BLACKBOARD          => BLACKBOARD);

   else
```

```
    -- Put on event list

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   EVENT_LISTS,
                    ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                   COMMON_EVENTS).LIST,
     THIS_ITEM => MODULE_PTR);
     -- Put on degree_modules level of blackboard

     PUT_ON_DEGREE_MODULES_LEVEL
     (BLACKBOARD => BLACKBOARD,
      MODULE_PTR => MODULE_PTR,
      MODULE     => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => MODULE));

  end if;

-------------------------------------------------------------------
end CREATE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Get_degree_modules gets all modules associated with the current
-- degree from an external file
--
-------------------------------------------------------------------
procedure GET_DEGREE_MODULES
-------------------------------------------------------------------
(FROM_FILENAME : in     STANDARD.STRING;
 BLACKBOARD    : in out TIMETABLE_BLACKBOARD_PACKAGE.
                        TIMETABLE_BLACKBOARD.
                        BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  FROM_DEGREE_FILE : TEXT_IO.FILE_TYPE;
  MODULE           : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
  MODULE_PTR       : TIMETABLE_TYPES_PACKAGE.
                     TIMETABLE_NODE_PTR_TYPE;
  STUDENTS         : SYSTEM_TYPES_PACKAGE.DYNAMIC_STRING.STRING;
-------------------------------------------------------------------
begin -- Get_degree_modules

  TEXT_IO.OPEN
  (FILE => FROM_DEGREE_FILE,
   MODE => TEXT_IO.IN_FILE,
   NAME => FROM_FILENAME);

  while not TEXT_IO.END_OF_FILE(FROM_DEGREE_FILE)
  loop
```

```
      if not SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.IS_NULL(MODULE) then

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        CLEAR
        (THE_STRING => MODULE);

      end if;

      if not SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.IS_NULL(STUDENTS) then

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        CLEAR
        (THE_STRING => STUDENTS);

      end if;


      SYSTEM_TYPES_PACKAGE.
      GET
      (TOKEN     => MODULE,
       FROM_FILE => FROM_DEGREE_FILE);

      SYSTEM_TYPES_PACKAGE.
      GET
      (TOKEN     => STUDENTS,
       FROM_FILE => FROM_DEGREE_FILE);

      CREATE(MODULE     => MODULE,
             STUDENTS   => STUDENTS,
             BLACKBOARD => BLACKBOARD);


    end loop;

    TEXT_IO.CLOSE(FROM_DEGREE_FILE);

  ------------------------------------------------------------------
  end GET_DEGREE_MODULES;
  ------------------------------------------------------------------

------------------------------------------------------------------
end SYLLABUS_KS_PACKAGE;
------------------------------------------------------------------
```

Requirement KS

```
------------------------------------------------------------------------
--
-- Unit      : REQUIREMENT_KS_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 12 January 1992
-- Function  : This package provides the blackboard transformation
--             from the degree_modules level to the
--             module_requirement level
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     RULE_BASE_TYPES_PACKAGE,
     GENERIC_RULE_BASE_INFERENCE_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
package REQUIREMENT_KS_PACKAGE is
------------------------------------------------------------------------

  procedure PROCESS_EVENT
  (MODULE_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD : in out TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_TYPE);

------------------------------------------------------------------------
end REQUIREMENT_KS_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : REQUIREMENT_KS_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 15 February 1992
-- Function  : This package provides the blackboard transformation
--             from the degree_modules level to the
--             module_requirements level
--
-------------------------------------------------------------------
package body REQUIREMENT_KS_PACKAGE is
-------------------------------------------------------------------

   Integrate a rule abstract knowledge type

   package RULE_BASE_PACKAGE is new
   GENERIC_RULE_BASE_INFERENCE_PACKAGE
   (RULE_BASES_FILENAME => "plmreqfile.list",
    FACT_COMPOSITE_TYPE => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_RECORD,
    FACT_PTR_TYPE       => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE,
    "<"                 => TIMETABLE_TYPES_PACKAGE."<",
    ">"                 => TIMETABLE_TYPES_PACKAGE.">",
    IS_EQUAL            => TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
    IS_LESS_THAN        => TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
    IS_GREATER_THAN     => TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
    PUT                 => TIMETABLE_TYPES_PACKAGE.PUT);

  -----------------------------------------------------------------
  --
  -- Allocate_activity adds the appropriate activity to the
  -- appropriate degree activity list
  --
  -----------------------------------------------------------------
  procedure ALLOCATE_REQUIREMENT
  -----------------------------------------------------------------
  (ACTIVITY     : in     TIMETABLE_TYPES_PACKAGE.ACTIVITY_TYPE;
   MODULE_PTR   : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   MODULE_CODE  : in     STANDARD.CHARACTER;
   RULE_PTR     : in     RULE_BASE_TYPES_PACKAGE.
                         RULE_BASE_NODE_PTR_TYPE;
   BLACKBOARD   : in out TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_TYPE) is
  -----------------------------------------------------------------
    NEW_REQUIREMENT_PTR : TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_NODE_PTR_TYPE;
    -------------------------------------------------------------
    MODULE_CODE_ERROR : exception;
  -----------------------------------------------------------------
  begin

    -- Generate new requirement

    NEW_REQUIREMENT_PTR := new TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_NODE_RECORD
                               (TIMETABLE_TYPES_PACKAGE.
```

```
                                MODULE_REQUIREMENT_KIND);

-- Set module name
SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY(FROM_THE_STRING => MODULE_PTR.FACT,
     TO_THE_STRING   => NEW_REQUIREMENT_PTR.FACT);

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
APPEND
(THE_SUBSTRING => "(" &
                  TIMETABLE_TYPES_PACKAGE.
                  ACTIVITY_TYPE'IMAGE(ACTIVITY)(1) & ")",
 TO_THE_STRING => NEW_REQUIREMENT_PTR.FACT);

-- Set fused data

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY(FROM_THE_STRING => NEW_REQUIREMENT_PTR.FACT,
     TO_THE_STRING   => NEW_REQUIREMENT_PTR.FUSED_FACT);

-- Copy students from pevious level

NEW_REQUIREMENT_PTR.STUDENTS := MODULE_PTR.STUDENTS;

-- Connect activity to the rule that created it

NEW_REQUIREMENT_PTR.RULE_PTR := RULE_PTR;

-- Join module supports to new requirement

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF(LIST      => MODULE_PTR.SUPPORTS,
               THIS_ITEM => NEW_REQUIREMENT_PTR);

-- Join new requirement supporters to module

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF(LIST      => NEW_REQUIREMENT_PTR.SUPPORTERS,
               THIS_ITEM => MODULE_PTR);

-- Add new requirement to blackboard module requirement

case MODULE_CODE is

  when 'E' =>

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST      => TIMETABLE_BLACKBOARD_PACKAGE.
                  TIMETABLE_BLACKBOARD.
                  BLACKBOARD_ITEM
                  (BLACKBOARD,
                   TIMETABLE_TYPES_PACKAGE.MODULE_REQUIREMENTS,
                   TIMETABLE_TYPES_PACKAGE.ESE).LIST,
```

```
        THIS_ITEM => NEW_REQUIREMENT_PTR);

    when 'I' =>

      TIMETABLE_TYPES_PACKAGE.
      TIMETABLE_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => TIMETABLE_BLACKBOARD_PACKAGE.
                    TIMETABLE_BLACKBOARD.
                    BLACKBOARD_ITEM
                    (BLACKBOARD,
                     TIMETABLE_TYPES_PACKAGE.MODULE_REQUIREMENTS,
                     TIMETABLE_TYPES_PACKAGE.IT).LIST,
       THIS_ITEM => NEW_REQUIREMENT_PTR);

    when others =>

      raise MODULE_CODE_ERROR;

  end case;

  -- Add new activity to event list

  TIMETABLE_TYPES_PACKAGE.
  TIMETABLE_LIST_PACKAGE.
  PUT_ON_BACK_OF(LIST      => TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_ITEM
                              (BLACKBOARD,
                               TIMETABLE_TYPES_PACKAGE.
                               EVENT_LISTS,
                               TIMETABLE_TYPES_PACKAGE.
                               COMMON_EVENTS).LIST,
                 THIS_ITEM => NEW_REQUIREMENT_PTR);
-------------------------------------------------------------------
exception

  when  MODULE_CODE_ERROR =>

    TEXT_IO.PUT_LINE
    ("Exception  MODULE_CODE_ERROR in REQUIREMENT_KS_PACKAGE" &
     " at ALLOCATE_REQUIREMENT");
-------------------------------------------------------------------
end ALLOCATE_REQUIREMENT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Find the lecture, tutorial and practical requirement for this
-- module
--
-------------------------------------------------------------------
procedure FIND
(REQUIREMENT      : in out TIMETABLE_TYPES_PACKAGE.
                           MODULE_REQUIREMENT_TYPE;
 REQUIREMENT_RULE : in     STRING) is
-------------------------------------------------------------------
  REQUIREMENT_RULE_LENGTH : NATURAL := REQUIREMENT_RULE'LENGTH;
  REQUIREMENT_1           : TIMETABLE_TYPES_PACKAGE.
```

```
                                ACTIVITY_TYPE;
  REQUIREMENT_2               : TIMETABLE_TYPES_PACKAGE.
                                ACTIVITY_TYPE;
  REQUIREMENT_3               : TIMETABLE_TYPES_PACKAGE.
                                ACTIVITY_TYPE;
-------------------------------------------------------------------
begin
  if  REQUIREMENT_RULE_LENGTH <= 17 then

    REQUIREMENT_1 := TIMETABLE_TYPES_PACKAGE.
                     ACTIVITY_TYPE'VALUE
                     (SYSTEM_TYPES_PACKAGE.
                      GET_FIELD(NUMBER => 2,
                      FROM => REQUIREMENT_RULE));
    REQUIREMENT(REQUIREMENT_1) := TRUE;

  elsif REQUIREMENT_RULE_LENGTH <= 26 then

    REQUIREMENT_1 := TIMETABLE_TYPES_PACKAGE.
                     ACTIVITY_TYPE'VALUE
                     (SYSTEM_TYPES_PACKAGE.
                      GET_FIELD(NUMBER => 2,
                      FROM => REQUIREMENT_RULE));
    REQUIREMENT_2 := TIMETABLE_TYPES_PACKAGE.
                     ACTIVITY_TYPE'VALUE
                     (SYSTEM_TYPES_PACKAGE.
                      GET_FIELD(NUMBER => 3,
                      FROM => REQUIREMENT_RULE));
    REQUIREMENT(REQUIREMENT_1) := TRUE;
    REQUIREMENT(REQUIREMENT_2) := TRUE;

  else

    REQUIREMENT(TIMETABLE_TYPES_PACKAGE.LECTURE) := TRUE;
    REQUIREMENT(TIMETABLE_TYPES_PACKAGE.TUTORIAL) := TRUE;

    REQUIREMENT_3 := TIMETABLE_TYPES_PACKAGE.
                     ACTIVITY_TYPE'VALUE
                     (SYSTEM_TYPES_PACKAGE.
                      GET_FIELD(NUMBER => 4,
                      FROM => REQUIREMENT_RULE));
    REQUIREMENT(REQUIREMENT_3) := TRUE;

  end if;
-------------------------------------------------------------------
end FIND;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Fire_rules fires all rules on the current agenda
--
-------------------------------------------------------------------
procedure FIRE_RULES
-------------------------------------------------------------------
(AGENDA      : in out RULE_BASE_TYPES_PACKAGE.
                      AGENDA_LIST_PACKAGE.
                      LIST_TYPE;
 MODULE_PTR  : in     TIMETABLE_TYPES_PACKAGE.
```

```
                        TIMETABLE_NODE_PTR_TYPE;
 MODULE_CODE : in       CHARACTER;
 BLACKBOARD  : in out TIMETABLE_BLACKBOARD_PACKAGE.
                        TIMETABLE_BLACKBOARD.
                        BLACKBOARD_TYPE
) is
-------------------------------------------------------------------
 MODULE_REQUIREMENT : TIMETABLE_TYPES_PACKAGE.
                      MODULE_REQUIREMENT_TYPE :=
                      (TIMETABLE_TYPES_PACKAGE.LECTURE   => FALSE,
                       TIMETABLE_TYPES_PACKAGE.TUTORIAL  => FALSE,
                       TIMETABLE_TYPES_PACKAGE.CAROUSEL  => FALSE,
                       TIMETABLE_TYPES_PACKAGE.PRACTICAL => FALSE)
                      ;

 RULE_PTR           : RULE_BASE_TYPES_PACKAGE.
                      RULE_BASE_NODE_PTR_TYPE;
-------------------------------------------------------------------
begin

  while not RULE_BASE_TYPES_PACKAGE.
            AGENDA_LIST_PACKAGE.
            IS_EMPTY(AGENDA)
  loop

    -- Get the rule

    RULE_BASE_TYPES_PACKAGE.
    AGENDA_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => AGENDA,
     THIS_ITEM => RULE_PTR);

    -- Rule action field format
    -- 12345678901234567890123456
    -- provide_all_activities
    -- provide_lecture
    -- provide_tutorial
    -- provide_practical
    -- provide_lecture_tutorial
    -- provide_lecture_practical
    -- provide_tutorial_practical

    -- Find requirement

    FIND(REQUIREMENT      => MODULE_REQUIREMENT,
         REQUIREMENT_RULE => SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             SUBSTRING_OF
                             (THE_STRING => RULE_PTR.
                                            ACTION.
                                            ACTION_TO_DO));

    -- Allocate the requirement

    for ACTIVITY in MODULE_REQUIREMENT'FIRST ..
                    MODULE_REQUIREMENT'LAST
    loop

      if MODULE_REQUIREMENT(ACTIVITY) then
```

```
          ALLOCATE_REQUIREMENT
          (ACTIVITY    => ACTIVITY,
           MODULE_PTR  => MODULE_PTR,
           MODULE_CODE => MODULE_CODE,
           RULE_PTR    => RULE_PTR,
           BLACKBOARD  => BLACKBOARD);

        end if;

      end loop;

    end loop;

  ------------------------------------------------------------------
  end FIRE_RULES;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  --
  -- Process_event selects the appropriate knowledge-base to process
  --  the current event
  --
  ------------------------------------------------------------------
  procedure PROCESS_EVENT
  ------------------------------------------------------------------
  (MODULE_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD : in out TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_TYPE) is
  ------------------------------------------------------------------
    MODULE_CODE : CHARACTER := SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               ITEM_OF(MODULE_PTR.FACT, 1);

    AGENDA      : RULE_BASE_TYPES_PACKAGE.
                  AGENDA_LIST_PACKAGE.
                  LIST_TYPE;
    ------------------------------------------------------------------
    MODULE_CODE_ERROR : exception;
    NON_EVENT         : exception;
  ------------------------------------------------------------------
  begin

    RULE_BASE_PACKAGE.
    INFERENCE
    (RULE_BASE => SYSTEM_TYPES_PACKAGE.
                  GET_FIELD
                  (NUMBER => 1,
                   FROM   => SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             SUBSTRING_OF
                             (THE_STRING => MODULE_PTR.FACT)),
     FACT_PTR  => MODULE_PTR,
     AGENDA    => AGENDA);

    if not RULE_BASE_TYPES_PACKAGE.
           AGENDA_LIST_PACKAGE.
```

```
          IS_EMPTY
          (AGENDA) then

      FIRE_RULES
      (AGENDA      => AGENDA,
       MODULE_PTR  => MODULE_PTR,
       MODULE_CODE => MODULE_CODE,
       BLACKBOARD  => BLACKBOARD);

    else

      raise NON_EVENT;

    end if;
  ------------------------------------------------------------------
  exception
    when MODULE_CODE_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception MODULE_CODE_ERROR in REQUIREMENT_KS_PACKAGE" &
       " at PROCESS_EVENT -> ");

    when NON_EVENT =>

      TEXT_IO.PUT_LINE
      ("Exception NON_EVENT in REQUIREMENT_KS_PACKAGE at " &
       "PROCESS_EVENT -> ");

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      PUT(THE_STRING => MODULE_PTR.FACT);

      TEXT_IO.NEW_LINE;
  ------------------------------------------------------------------
  end PROCESS_EVENT;
  ------------------------------------------------------------------

------------------------------------------------------------------
end REQUIREMENT_KS_PACKAGE;
------------------------------------------------------------------
```

Common KS

```
------------------------------------------------------------------------
--
-- Unit      : COMMON_KS_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 17 January 1992
-- Function  : This package provides the blackboard transformation
--             from the module_requirements level to the
--             common_module_requirements level
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     RULE_BASE_TYPES_PACKAGE,
     GENERIC_RULE_BASE_INFERENCE_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
package COMMON_KS_PACKAGE is
------------------------------------------------------------------------

  procedure PROCESS_EVENT
  (MODULE_REQUIREMENT_PTR : in     TIMETABLE_TYPES_PACKAGE.
                                   TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD             : in out TIMETABLE_BLACKBOARD_PACKAGE.
                                   TIMETABLE_BLACKBOARD.
                                   BLACKBOARD_TYPE);

------------------------------------------------------------------------
end COMMON_KS_PACKAGE;
------------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Unit      : COMMON_KS_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 17 January 1992
-- Function  : This package provides the blackboard transformation
--             from the module_requirement level to the
--             common_module_requirement level
--
------------------------------------------------------------------
package body COMMON_KS_PACKAGE is
------------------------------------------------------------------

   package RULE_BASE_PACKAGE is new
   GENERIC_RULE_BASE_INFERENCE_PACKAGE
   (RULE_BASES_FILENAME => "plmcomfile.list",
    FACT_COMPOSITE_TYPE => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_RECORD,
    FACT_PTR_TYPE       => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE,
    "<"                 => TIMETABLE_TYPES_PACKAGE."<",
    ">"                 => TIMETABLE_TYPES_PACKAGE.">",
    IS_EQUAL            => TIMETABLE_TYPES_PACKAGE.IS_EQUAL,
    IS_LESS_THAN        => TIMETABLE_TYPES_PACKAGE.IS_LESS_THAN,
    IS_GREATER_THAN     => TIMETABLE_TYPES_PACKAGE.IS_GREATER_THAN,
    PUT                 => TIMETABLE_TYPES_PACKAGE.PUT);

  ----------------------------------------------------------------
  --
  -- Allocate_new_common_module creates a new entry at the common
  -- level of the blackboard
  --
  ----------------------------------------------------------------
  procedure ALLOCATE_NEW_COMMON_MODULE_REQUIREMENT
  ----------------------------------------------------------------
  ( MODULE_REQUIREMENT_PTR : in     TIMETABLE_TYPES_PACKAGE.
                                    TIMETABLE_NODE_PTR_TYPE;
    BLACKBOARD             : in out TIMETABLE_BLACKBOARD_PACKAGE.
                                    TIMETABLE_BLACKBOARD.
                                    BLACKBOARD_TYPE) is
    ----------------------------------------------------------------

    NEW_COMMON_MODULE_REQUIREMENT_PTR : TIMETABLE_TYPES_PACKAGE.
                                        TIMETABLE_NODE_PTR_TYPE;

  ----------------------------------------------------------------
  begin

    -- Generate new common module requirement

    NEW_COMMON_MODULE_REQUIREMENT_PTR :=
    new TIMETABLE_TYPES_PACKAGE.
        TIMETABLE_NODE_RECORD
        (TIMETABLE_TYPES_PACKAGE.
         MODULE_COMMON_REQUIREMENT_KIND);

    -- Set common requirement fact to module requirement fact

    SYSTEM_TYPES_PACKAGE.
```

```
DYNAMIC_STRING.
COPY
(FROM_THE_STRING => MODULE_REQUIREMENT_PTR.FACT,
 TO_THE_STRING   => NEW_COMMON_MODULE_REQUIREMENT_PTR.FACT);

-- Set fused fact to degree module name

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_STRING => MODULE_REQUIREMENT_PTR.FACT,
 TO_THE_STRING   => NEW_COMMON_MODULE_REQUIREMENT_PTR.
                    FUSED_FACT);

-- Record students from lower level

NEW_COMMON_MODULE_REQUIREMENT_PTR.STUDENTS :=
MODULE_REQUIREMENT_PTR.STUDENTS;

-- Join new MODULE supports to new common MODULE

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF
(LIST      => MODULE_REQUIREMENT_PTR.SUPPORTS,
 THIS_ITEM => NEW_COMMON_MODULE_REQUIREMENT_PTR);

-- Join new common requirement supporters to module requirement

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF
(LIST      => NEW_COMMON_MODULE_REQUIREMENT_PTR.SUPPORTERS,
 THIS_ITEM => MODULE_REQUIREMENT_PTR);

-- Add new common module requirement to blackboard common module
-- requirement level

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF
(LIST      => TIMETABLE_BLACKBOARD_PACKAGE.
              TIMETABLE_BLACKBOARD.
              BLACKBOARD_ITEM
              (BLACKBOARD  => BLACKBOARD,
               LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                              COMMON_MODULE_REQUIREMENTS,
               ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                              COMMON_MODULE_REQUIREMENT_LIST).
                 LIST,
 THIS_ITEM => NEW_COMMON_MODULE_REQUIREMENT_PTR);

-- Add new common module requirement to event list

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF
(LIST      => TIMETABLE_BLACKBOARD_PACKAGE.
              TIMETABLE_BLACKBOARD.
              BLACKBOARD_ITEM
```

```
                (BLACKBOARD  => BLACKBOARD,
                 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                EVENT_LISTS,
                 ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                PERIOD_EVENTS).
                                LIST,
   THIS_ITEM => NEW_COMMON_MODULE_REQUIREMENT_PTR);

-------------------------------------------------------------------
end ALLOCATE_NEW_COMMON_MODULE_REQUIREMENT;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Find a module on the common module list
--
-------------------------------------------------------------------
procedure FIND
-------------------------------------------------------------------
(COMMON_MODULE_REQUIREMENT_PTR :    out  TIMETABLE_TYPES_PACKAGE.
                                         TIMETABLE_NODE_PTR_TYPE;
 COMMON_MODULE_REQUIREMENT     : in       STANDARD.STRING;
 ON_COMMON_LIST                : in       TIMETABLE_TYPES_PACKAGE.
                                          TIMETABLE_LIST_PACKAGE.
                                          LIST_TYPE) is
  -----------------------------------------------------------------
  COMMON_REQUIREMENT_PTR : TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
  MODULE_REQUIREMENT_PTR : TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
  -----------------------------------------------------------------
  NOT_FOUND : exception;
-------------------------------------------------------------------
begin

  for COMMON_COUNT in 1 .. TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_LIST_PACKAGE.
                           LENGTH_OF(ON_COMMON_LIST)
  loop

    COMMON_REQUIREMENT_PTR := TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_LIST_PACKAGE.
                              ITEM_AT
                              (LIST        => ON_COMMON_LIST,
                               NODE_NUMBER => COMMON_COUNT);

    for MODULE_REQUIREMENT_COUNT in
        1 .. TIMETABLE_TYPES_PACKAGE.
             TIMETABLE_LIST_PACKAGE.
             LENGTH_OF
             (COMMON_REQUIREMENT_PTR.
              SUPPORTERS)
    loop

      MODULE_REQUIREMENT_PTR := TIMETABLE_TYPES_PACKAGE.
                                TIMETABLE_LIST_PACKAGE.
                                ITEM_AT
                                (LIST        =>
                                 COMMON_REQUIREMENT_PTR.
```

```
                                    SUPPORTERS,
                                    NODE_NUMBER =>
                                    MODULE_REQUIREMENT_COUNT);

      if COMMON_MODULE_REQUIREMENT = SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     SUBSTRING_OF
                                     (THE_STRING =>
                                      MODULE_REQUIREMENT_PTR.
                                      FACT) then

        COMMON_MODULE_REQUIREMENT_PTR := COMMON_REQUIREMENT_PTR;
        return;

      end if;

    end loop;

  end loop;

  raise NOT_FOUND;

exception

  when NOT_FOUND =>

    TEXT_IO.PUT
    ("Exception NOT_FOUND at FIND in COMMON_KS_PACKAGE -> ");
    TEXT_IO.PUT_LINE
    (COMMON_MODULE_REQUIREMENT);

-------------------------------------------------------------------
end FIND;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Make_common attaches the new MODULE event to an existing common
-- module
--
-------------------------------------------------------------------
procedure MAKE_COMMON
-------------------------------------------------------------------
(MODULE_REQUIREMENT_PTR     : in     TIMETABLE_TYPES_PACKAGE.
                                     TIMETABLE_NODE_PTR_TYPE;
 COMMON_MODULE_REQUIREMENT : in      STRING;
 BLACKBOARD                : in out TIMETABLE_BLACKBOARD_PACKAGE.
                                     TIMETABLE_BLACKBOARD.
                                     BLACKBOARD_TYPE) is
-------------------------------------------------------------------

  COMMON_MODULE_REQUIREMENT_PTR : TIMETABLE_TYPES_PACKAGE.
                                  TIMETABLE_NODE_PTR_TYPE;

  TEMP_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        STRING;
begin
```

```
   -- Find the existing common module which will be an existing
   -- requirement on the requirement level supporting an entry on
   -- the common level

  FIND
  (COMMON_MODULE_REQUIREMENT_PTR => COMMON_MODULE_REQUIREMENT_PTR,
   COMMON_MODULE_REQUIREMENT     => COMMON_MODULE_REQUIREMENT,
   ON_COMMON_LIST                => TIMETABLE_BLACKBOARD_PACKAGE.
                                    TIMETABLE_BLACKBOARD.
                                    BLACKBOARD_ITEM
                                    (BLACKBOARD  => BLACKBOARD,
                                     LEVEL_INDEX =>
                                     TIMETABLE_TYPES_PACKAGE.
                                     COMMON_MODULE_REQUIREMENTS,
                                     ITEM_INDEX  =>
                                     TIMETABLE_TYPES_PACKAGE.
                                    COMMON_MODULE_REQUIREMENT_LIST).
                                    LIST);

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY(FROM_THE_STRING => MODULE_REQUIREMENT_PTR.
                           FACT,
        TO_THE_STRING   => TEMP_DYNAMIC_STRING);

   if not SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING_LIST_PACKAGE.
          IS_IN
          (LIST => COMMON_MODULE_REQUIREMENT_PTR.
                   LIST_OF_COMMON_MODULES,
           ITEM => TEMP_DYNAMIC_STRING) then

   -- Not already there so add new common module to list of common
   -- modules

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING_LIST_PACKAGE.
   PUT_ON_FRONT_OF
   (LIST      => COMMON_MODULE_REQUIREMENT_PTR.
                 LIST_OF_COMMON_MODULES,
    THIS_ITEM => TEMP_DYNAMIC_STRING);

         -- Add students from lower level if from different degree

         if SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            ITEM_OF
            (THE_STRING => COMMON_MODULE_REQUIREMENT_PTR.FACT,
             AT_THE_POSITION => 1)

            /=

            SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            ITEM_OF
            (THE_STRING => MODULE_REQUIREMENT_PTR.FACT,
             AT_THE_POSITION => 1) then


           COMMON_MODULE_REQUIREMENT_PTR.STUDENTS :=
```

```
            COMMON_MODULE_REQUIREMENT_PTR.STUDENTS +
            MODULE_REQUIREMENT_PTR.STUDENTS;

        end if;

    -- Fuse the fact from the module requirement into the common
    -- requirement

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    PREPEND
    (THE_SUBSTRING => "_",
     TO_THE_STRING => COMMON_MODULE_REQUIREMENT_PTR.FUSED_FACT);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    PREPEND
    (THE_STRING    => MODULE_REQUIREMENT_PTR.FACT,
     TO_THE_STRING => COMMON_MODULE_REQUIREMENT_PTR.FUSED_FACT);

    -- Increment the module count

    COMMON_MODULE_REQUIREMENT_PTR.NUMBER_OF_COMMON_MODULES :=
    COMMON_MODULE_REQUIREMENT_PTR.NUMBER_OF_COMMON_MODULES + 1;

    -- Make common by adding new requirement ptr to same supporter
    -- list as existing common reqirement

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST      => COMMON_MODULE_REQUIREMENT_PTR.
                  SUPPORTERS,
     THIS_ITEM => MODULE_REQUIREMENT_PTR);

    -- connect new requirement to support existing common
    -- requirement

     TIMETABLE_TYPES_PACKAGE.
     TIMETABLE_LIST_PACKAGE.
     PUT_ON_BACK_OF
     (LIST      => MODULE_REQUIREMENT_PTR.
                   SUPPORTS,
      THIS_ITEM => COMMON_MODULE_REQUIREMENT_PTR);

   end if;

-------------------------------------------------------------------
end  MAKE_COMMON;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Fire_rules
--
-------------------------------------------------------------------
procedure FIRE_RULES
-------------------------------------------------------------------
(AGENDA                    : in out RULE_BASE_TYPES_PACKAGE.
```

```
                                    AGENDA_LIST_PACKAGE.
                                    LIST_TYPE;
 MODULE_REQUIREMENT_PTR  : in       TIMETABLE_TYPES_PACKAGE.
                                    TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD              : in out   TIMETABLE_BLACKBOARD_PACKAGE.
                                    TIMETABLE_BLACKBOARD.
                                    BLACKBOARD_TYPE) is
 ------------------------------------------------------------------
 RULE_PTR : RULE_BASE_TYPES_PACKAGE.
            RULE_BASE_NODE_PTR_TYPE;
------------------------------------------------------------------
begin

  if RULE_BASE_TYPES_PACKAGE.
     AGENDA_LIST_PACKAGE.
     IS_EMPTY
     (LIST => AGENDA) then

    -- Create a new module at the common level
    ALLOCATE_NEW_COMMON_MODULE_REQUIREMENT
    (MODULE_REQUIREMENT_PTR => MODULE_REQUIREMENT_PTR,
     BLACKBOARD             => BLACKBOARD);

  else

    while not RULE_BASE_TYPES_PACKAGE.
              AGENDA_LIST_PACKAGE.
              IS_EMPTY
              (AGENDA)
    loop

      RULE_BASE_TYPES_PACKAGE.
      AGENDA_LIST_PACKAGE.
      GET_FROM_FRONT_OF
      (LIST      => AGENDA,
       THIS_ITEM => RULE_PTR);

        -- The rule action format
        --|  1 | 2  | 3 | 4  |  5   |
        -- make_E101(L)_and_I111(L)_common

      if SYSTEM_TYPES_PACKAGE.
        GET_FIELD
        (2,
         SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         SUBSTRING_OF
         (THE_STRING => RULE_PTR.ACTION.ACTION_TO_DO)) =

         SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         SUBSTRING_OF(THE_STRING => MODULE_REQUIREMENT_PTR.FACT)

       then

         MAKE_COMMON
         (MODULE_REQUIREMENT_PTR     => MODULE_REQUIREMENT_PTR,
          COMMON_MODULE_REQUIREMENT => SYSTEM_TYPES_PACKAGE.
                                       GET_FIELD
                                       (4,
```

```
                                        SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.
                                        SUBSTRING_OF
                                        (THE_STRING =>
                                         RULE_PTR.
                                         ACTION.
                                         ACTION_TO_DO)),
          BLACKBOARD                 => BLACKBOARD);

      else

         MAKE_COMMON
         (MODULE_REQUIREMENT_PTR     => MODULE_REQUIREMENT_PTR,
          COMMON_MODULE_REQUIREMENT  => SYSTEM_TYPES_PACKAGE.
                                        GET_FIELD
                                        (2,
                                         SYSTEM_TYPES_PACKAGE.
                                         DYNAMIC_STRING.
                                         SUBSTRING_OF
                                         (THE_STRING =>
                                          RULE_PTR.
                                          ACTION.
                                          ACTION_TO_DO)),
          BLACKBOARD                 => BLACKBOARD);

      end if;

    end loop;

  end if;

-------------------------------------------------------------------
end FIRE_RULES;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PROCESS_EVENT
-------------------------------------------------------------------
(MODULE_REQUIREMENT_PTR : in     TIMETABLE_TYPES_PACKAGE.
                                 TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD             : in out TIMETABLE_BLACKBOARD_PACKAGE.
                                 TIMETABLE_BLACKBOARD.
                                 BLACKBOARD_TYPE) is
  -----------------------------------------------------------------

  AGENDA : RULE_BASE_TYPES_PACKAGE.
           AGENDA_LIST_PACKAGE.LIST_TYPE;

-------------------------------------------------------------------
begin

  RULE_BASE_PACKAGE.
  INFERENCE
  (RULE_BASE => SYSTEM_TYPES_PACKAGE.
                GET_FIELD
                (NUMBER => 1,
                 FROM   => SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           SUBSTRING_OF
```

```
                                (THE_STRING => MODULE_REQUIREMENT_PTR.
                                               FACT)),
     FACT_PTR   => MODULE_REQUIREMENT_PTR,
     AGENDA     => AGENDA);


    FIRE_RULES
    (AGENDA                     => AGENDA,
     MODULE_REQUIREMENT_PTR     => MODULE_REQUIREMENT_PTR,
     BLACKBOARD                 => BLACKBOARD);

  ------------------------------------------------------------------------
  end PROCESS_EVENT;
  ------------------------------------------------------------------------

------------------------------------------------------------------------
end COMMON_KS_PACKAGE;
------------------------------------------------------------------------
```

Activity KS

```
------------------------------------------------------------------------
--
-- Unit      : ACTIVITY_KS_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 12 January 1992
-- Function  : This package provides the blackboard transformation
--             from the degree_modules level to the degree_activities
--             level
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     RULE_BASE_TYPES_PACKAGE,
     GENERIC_RULE_BASE_INFERENCE_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
package ACTIVITY_KS_PACKAGE is
------------------------------------------------------------------------

  procedure PROCESS_EVENT
  (COMMON_MODULE_PTR : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD        : in out TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_TYPE);

------------------------------------------------------------------------
end ACTIVITY_KS_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : ACTIVITY_KS_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             CRanfield University, RMCS, Shrivenham
-- Date      : 12 January 1992
-- Function  : This package provides the blackboard transformation
--             from the
--             common_modules level to the degree_activities level
--
-------------------------------------------------------------------
package body ACTIVITY_KS_PACKAGE is
-------------------------------------------------------------------

   Integrate a rule abstract knowledge type

   package RULE_BASE_PACKAGE is new
   GENERIC_RULE_BASE_INFERENCE_PACKAGE
   (RULE_BASES_FILENAME => "plmactfile.list",
    FACT_COMPOSITE_TYPE => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_RECORD,
    FACT_PTR_TYPE       => TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE,
    "<"                 => TIMETABLE_TYPES_PACKAGE.
                           "<",
    ">"                 => TIMETABLE_TYPES_PACKAGE.
                           ">",
    IS_EQUAL            => TIMETABLE_TYPES_PACKAGE.
                           IS_EQUAL,
    IS_LESS_THAN        => TIMETABLE_TYPES_PACKAGE.
                           IS_LESS_THAN,
    IS_GREATER_THAN     => TIMETABLE_TYPES_PACKAGE.
                           IS_GREATER_THAN,
    PUT                 => TIMETABLE_TYPES_PACKAGE.PUT);

  -----------------------------------------------------------------
  --
  -- Preferece extracts the day and period preference for an
  -- activity
  --
  -----------------------------------------------------------------
  function PREFERENCE
  (NUMBER : in POSITIVE;
   RULE_PTR : in RULE_BASE_TYPES_PACKAGE.
                 RULE_BASE_NODE_PTR_TYPE)
  return STANDARD.STRING is
  -----------------------------------------------------------------
    OFFSET : constant POSITIVE := 9;
  -----------------------------------------------------------------
  begin
    if SYSTEM_TYPES_PACKAGE.
       IS_FIELD_AT(OFFSET + NUMBER,
                   SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   SUBSTRING_OF(THE_STRING => RULE_PTR.
                                              ACTION.
                                              ACTION_TO_DO)) then
      return SYSTEM_TYPES_PACKAGE.
             GET_FIELD(NUMBER => OFFSET + NUMBER,
                       FROM   => SYSTEM_TYPES_PACKAGE.
```

```
                                    DYNAMIC_STRING.
                                    SUBSTRING_OF
                                    (THE_STRING => RULE_PTR.
                                                   ACTION.
                                                   ACTION_TO_DO));
  else
    return "";
  end if;

-------------------------------------------------------------------
end PREFERENCE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Allocate _activity adds the appropriate activity to the
-- appropriate degree activity list
--
-------------------------------------------------------------------
procedure ALLOCATE_ACTIVITY
-------------------------------------------------------------------
(NUMBER              : in     STANDARD.POSITIVE;
 COMMON_MODULE_PTR   : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_NODE_PTR_TYPE;
 ACTIVITY_CODE       : in     STANDARD.
                              CHARACTER;
 FREQUENCY           : in     SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 NUMBER_OF_PERIODS   : in     STANDARD.
                              POSITIVE;
 PREFERENCE          : in     STANDARD.STRING;
 RULE_PTR            : in     RULE_BASE_TYPES_PACKAGE.
                              RULE_BASE_NODE_PTR_TYPE;
 BLACKBOARD          : in out TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  NEW_ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                     TIMETABLE_NODE_PTR_TYPE;
  -----------------------------------------------------------------
  ACTIVITY_CODE_ERROR : exception;
-------------------------------------------------------------------
begin

  -- Generate new activity

  NEW_ACTIVITY_PTR := new TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_NODE_RECORD
                          (TIMETABLE_TYPES_PACKAGE.
                           MODULE_ACTIVITY_KIND);

  -- Set next action to be taken

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "Allocate Staff",
   TO_THE_STRING      => NEW_ACTIVITY_PTR.
```

```
                          ACTION);

   -- Set module name

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY(FROM_THE_STRING => COMMON_MODULE_PTR.FACT,
        TO_THE_STRING   => NEW_ACTIVITY_PTR.FACT);

   -- Record students from lower level

   NEW_ACTIVITY_PTR.STUDENTS := COMMON_MODULE_PTR.STUDENTS;

   -- Set the preference

   if PREFERENCE'LENGTH > 0 then

     NEW_ACTIVITY_PTR.HAS_PREFERENCE := TRUE;

     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     COPY
     (FROM_THE_SUBSTRING => (PREFERENCE
                             (PREFERENCE'FIRST ..
                              PREFERENCE'LAST - 2)),
      TO_THE_STRING      => NEW_ACTIVITY_PTR.DAY_PREFERENCE);

     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     COPY
     (FROM_THE_SUBSTRING => (PREFERENCE
                             (PREFERENCE'LAST..PREFERENCE'LAST)),
      TO_THE_STRING      => NEW_ACTIVITY_PTR.PERIOD_PREFERENCE);

   else

     NEW_ACTIVITY_PTR.HAS_PREFERENCE := FALSE;

   end if;

   -- Set fused data

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY(FROM_THE_STRING => COMMON_MODULE_PTR.FUSED_FACT,
        TO_THE_STRING   => NEW_ACTIVITY_PTR.FUSED_FACT);

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   APPEND
   (THE_SUBSTRING =>
      "_" &
      STANDARD.POSITIVE'IMAGE(NUMBER_OF_PERIODS)
                              (2 .. STANDARD.POSITIVE'IMAGE
                                    (NUMBER_OF_PERIODS)'LENGTH) &
      "_" &
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      SUBSTRING_OF
      (THE_STRING    => FREQUENCY) & "_" & PREFERENCE,
```

```
    TO_THE_STRING => NEW_ACTIVITY_PTR.FUSED_FACT);

-- Copy common activities

for I in 1 .. SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING_LIST_PACKAGE.
              LENGTH_OF
              (LIST => COMMON_MODULE_PTR.
                       LIST_OF_COMMON_MODULES)
loop
  declare

    TEMP : SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           STRING;

  begin

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_STRING => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING_LIST_PACKAGE.
                        ITEM_AT
                        (LIST        => COMMON_MODULE_PTR.
                                        LIST_OF_COMMON_MODULES,
                         NODE_NUMBER => I),
     TO_THE_STRING   => TEMP);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST => NEW_ACTIVITY_PTR.
             LIST_OF_COMMON_ACTIVITIES,
     THIS_ITEM => TEMP);

  end;

end loop;

-- Set activity value

NEW_ACTIVITY_PTR.ACTIVITY := ACTIVITY_CODE;

-- Week frequency

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_STRING => FREQUENCY,
 TO_THE_STRING   => NEW_ACTIVITY_PTR.FREQUENCY);

-- Number of periods

NEW_ACTIVITY_PTR.NUMBER_OF_PERIODS := NUMBER_OF_PERIODS;

-- Connect activity to the rule that created it

NEW_ACTIVITY_PTR.RULE_PTR := RULE_PTR;
```

```
-- Join common module supports to new degree activity

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF(LIST       => COMMON_MODULE_PTR.SUPPORTS,
               THIS_ITEM  => NEW_ACTIVITY_PTR);

-- Join new activity supporters to common module

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
PUT_ON_BACK_OF(LIST       => NEW_ACTIVITY_PTR.SUPPORTERS,
               THIS_ITEM  => COMMON_MODULE_PTR);
-- Add new activity to blackboard degree activities

case ACTIVITY_CODE is

  when 'L' =>

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DEGREE_ACTIVITIES,
                    ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                   LECTURES).
                                   LIST,
     THIS_ITEM => NEW_ACTIVITY_PTR);

  when 'T' =>

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DEGREE_ACTIVITIES,
                    ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                   TUTORIALS).
                                   LIST,
     THIS_ITEM => NEW_ACTIVITY_PTR);

  when 'C' =>

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DEGREE_ACTIVITIES,
```

```
                        ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                       CAROUSEL).
                                       LIST,
          THIS_ITEM => NEW_ACTIVITY_PTR);

      when 'P' =>

        TIMETABLE_TYPES_PACKAGE.
        TIMETABLE_LIST_PACKAGE.
        PUT_ON_BACK_OF
        (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_ITEM
                       (BLACKBOARD  => BLACKBOARD,
                        LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                       DEGREE_ACTIVITIES,
                        ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                       PRACTICALS).
                                       LIST,
          THIS_ITEM => NEW_ACTIVITY_PTR);

      when others =>

        raise ACTIVITY_CODE_ERROR;

    end case;

    -- Add new activity to event list

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF(LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                                 TIMETABLE_BLACKBOARD.
                                 BLACKBOARD_ITEM
                                 (BLACKBOARD  => BLACKBOARD,
                                  LEVEL_INDEX =>
                                  TIMETABLE_TYPES_PACKAGE.
                                  EVENT_LISTS,
                                  ITEM_INDEX  =>
                                  TIMETABLE_TYPES_PACKAGE.
                                  PERIOD_EVENTS).
                                  LIST,
                   THIS_ITEM => NEW_ACTIVITY_PTR);.
------------------------------------------------------------------
exception
  when  ACTIVITY_CODE_ERROR =>

    TEXT_IO.PUT_LINE
    ("Exception  ACTIVITY_CODE_ERROR in TUTOR_KS_PACKAGE" &
     " at ALLOCATE_ACTIVITY");
------------------------------------------------------------------
end ALLOCATE_ACTIVITY;
------------------------------------------------------------------


------------------------------------------------------------------
--
-- Fire_rules fires all rules on the current agenda
--
------------------------------------------------------------------
```

```
procedure FIRE_RULES
-------------------------------------------------------------------
(AGENDA              : in out RULE_BASE_TYPES_PACKAGE.
                              AGENDA_LIST_PACKAGE.
                              LIST_TYPE;
 COMMON_MODULE_PTR   : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD          : in out TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  ACTIVITY_CODE      : STANDARD.CHARACTER;
  NUMBER_PER_WEEK    : STANDARD.POSITIVE;
  FREQUENCY          : SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING.
                       STRING;
  NUMBER_OF_PERIODS  : STANDARD.POSITIVE;
  RULE_PTR           : RULE_BASE_TYPES_PACKAGE.
                       RULE_BASE_NODE_PTR_TYPE;
-------------------------------------------------------------------
begin

  while not RULE_BASE_TYPES_PACKAGE.
            AGENDA_LIST_PACKAGE.
            IS_EMPTY(AGENDA)
  loop

    -- Get the rule

    RULE_BASE_TYPES_PACKAGE.
    AGENDA_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => AGENDA,
     THIS_ITEM => RULE_PTR);

    -- Rule action field format
    --|   1    |2|3|4|.5    |   6    |7 |  8  | 9  |  10 |  11
    -- allocate_2_/_1_period_Lectures_in_weeks_1-11_tue-3_thu-3

    -- Activity

    ACTIVITY_CODE :=
      SYSTEM_TYPES_PACKAGE.
      GET_FIELD
      (6,
       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       SUBSTRING_OF
       (THE_STRING => RULE_PTR.
        ACTION.
        ACTION_TO_DO))(SYSTEM_TYPES_PACKAGE.
                       GET_FIELD
                       (6,
                        SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => RULE_PTR.
                                       ACTION.
                                       ACTION_TO_DO))'FIRST);
```

```
-- Number per week

NUMBER_PER_WEEK := POSITIVE'VALUE
                   (SYSTEM_TYPES_PACKAGE.
                    GET_FIELD
                    (2,
                     SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     SUBSTRING_OF
                     (THE_STRING => RULE_PTR.
                                    ACTION.
                                    ACTION_TO_DO)));

-- Frequency in term

if not SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       IS_NULL
       (THE_STRING => FREQUENCY) then

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  CLEAR
  (THE_STRING => FREQUENCY);

end if;

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                       GET_FIELD
                       (9,
                        SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => RULE_PTR.
                                       ACTION.
                                       ACTION_TO_DO)),
 TO_THE_STRING => FREQUENCY);

-- Number of periods

NUMBER_OF_PERIODS := POSITIVE'VALUE
                     (SYSTEM_TYPES_PACKAGE.
                      GET_FIELD
                      (4,
                       SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING.
                       SUBSTRING_OF
                       (THE_STRING => RULE_PTR.
                                      ACTION.
                                      ACTION_TO_DO)));

-- Allocate the activities required

for COUNT in 1 .. NUMBER_PER_WEEK
loop

  ALLOCATE_ACTIVITY
```

```
        (NUMBER              => COUNT,
         COMMON_MODULE_PTR   => COMMON_MODULE_PTR,
         ACTIVITY_CODE       => ACTIVITY_CODE,
         FREQUENCY           => FREQUENCY,
         NUMBER_OF_PERIODS   => NUMBER_OF_PERIODS,
         PREFERENCE          => PREFERENCE(COUNT, RULE_PTR),
         RULE_PTR            => RULE_PTR,
         BLACKBOARD          => BLACKBOARD);

      end loop;
    end loop;

------------------------------------------------------------------
end FIRE_RULES;
------------------------------------------------------------------


------------------------------------------------------------------
--
-- Process_event selects the appropriate knowledge-base to process
-- the current event
--
------------------------------------------------------------------
procedure PROCESS_EVENT
------------------------------------------------------------------
(COMMON_MODULE_PTR : in     TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD        : in out TIMETABLE_BLACKBOARD_PACKAGE.
                            TIMETABLE_BLACKBOARD.
                            BLACKBOARD_TYPE) is
------------------------------------------------------------------
  AGENDA              : RULE_BASE_TYPES_PACKAGE.
                        AGENDA_LIST_PACKAGE.
                        LIST_TYPE;
  ------------------------------------------------------------------
  NON_EVENT                 : exception;
------------------------------------------------------------------
begin

  RULE_BASE_PACKAGE.
  INFERENCE
  (RULE_BASE => SYSTEM_TYPES_PACKAGE.
                GET_FIELD
                (NUMBER => 1,
                 FROM   => SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           SUBSTRING_OF
                           (THE_STRING => COMMON_MODULE_PTR.
                                          FACT)),

   FACT_PTR => COMMON_MODULE_PTR,
   AGENDA   => AGENDA);

  if not RULE_BASE_TYPES_PACKAGE.
         AGENDA_LIST_PACKAGE.
         IS_EMPTY
         (AGENDA) then

    FIRE_RULES
    (AGENDA             => AGENDA,
```

```
        COMMON_MODULE_PTR   => COMMON_MODULE_PTR,
        BLACKBOARD          => BLACKBOARD);

    else

      raise NON_EVENT;

    end if;

  ------------------------------------------------------------------
  exception
    when NON_EVENT =>
      TEXT_IO.PUT("Exception NON_EVENT in TUTOR_KS_PACKAGE at " &
                  "PROCESS_EVENT -> ");

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      PUT(THE_STRING => COMMON_MODULE_PTR.FACT);

      TEXT_IO.NEW_LINE;
  ------------------------------------------------------------------
  end PROCESS_EVENT;
  ------------------------------------------------------------------

------------------------------------------------------------------
end ACTIVITY_KS_PACKAGE;
------------------------------------------------------------------
```

Staff

```
------------------------------------------------------------------------
--
-- Unit      : STAFF_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 21 JULY 1992
-- Function  : This package provides the operations to build a
--             STAFF Frame System
--
------------------------------------------------------------------------
with GENERIC_FRAME_BASE_PACKAGE,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TEXT_IO;
------------------------------------------------------------------------
package STAFF_PACKAGE is
------------------------------------------------------------------------

  -- Integrate a frame abstract knowledge type

  package STAFF_FRAME_BASE_PACKAGE is new GENERIC_FRAME_BASE_PACKAGE
  (SLOT_TYPE              => TIMETABLE_TYPES_PACKAGE.
                             STAFF_SLOT_TYPE,
   FACET_RECORD_TYPE      => TIMETABLE_TYPES_PACKAGE.
                             STAFF_FACET_RECORD_TYPE,
   FACET_RECORD_PTR_TYPE  => TIMETABLE_TYPES_PACKAGE.
                             STAFF_FACET_RECORD_PTR_TYPE,
   "<"                    => TIMETABLE_TYPES_PACKAGE.
                             "<",
   ">"                    => TIMETABLE_TYPES_PACKAGE.
                             ">",
   IS_EQUAL               => TIMETABLE_TYPES_PACKAGE.
                             IS_EQUAL,
   IS_LESS_THAN           => TIMETABLE_TYPES_PACKAGE.
                             IS_LESS_THAN,
   IS_GREATER_THAN        => TIMETABLE_TYPES_PACKAGE.
                             IS_GREATER_THAN,
   PUT                    => TIMETABLE_TYPES_PACKAGE.
                             PUT,
   GET                    => TIMETABLE_TYPES_PACKAGE.
                             GET);

  STAFF_FRAME_BASE_RECORD : STAFF_FRAME_BASE_PACKAGE.
                            FRAME_BASE_RECORD_TYPE;

  -- Define additional user application operations

  procedure MAKE_BUSY
  (LECTURER                : in     SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
   MODULE                  : in     SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
   DAY                     : in     TIMETABLE_TYPES_PACKAGE.
                                    DAY_TYPE;
   PERIOD                  : in     TIMETABLE_TYPES_PACKAGE.
```

```
                                    PERIOD_NUMBER_TYPE;
 WEEK_ARRAY              : in       TIMETABLE_TYPES_PACKAGE.
                                    WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD : in out STAFF_FRAME_BASE_PACKAGE.
                                    FRAME_BASE_RECORD_TYPE);

procedure MAKE_FREE
(LECTURER                : in       SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
 MODULE                  : in       SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
 DAY                     : in       TIMETABLE_TYPES_PACKAGE.
                                    DAY_TYPE;
 PERIOD                  : in       TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE;
 WEEK_ARRAY              : in       TIMETABLE_TYPES_PACKAGE.
                                    WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD : in out STAFF_FRAME_BASE_PACKAGE.
                                    FRAME_BASE_RECORD_TYPE);

function IS_FREE
(LECTURER                : in SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                STRING;
 DAY                     : in TIMETABLE_TYPES_PACKAGE.
                                DAY_TYPE;
 PERIOD                  : in TIMETABLE_TYPES_PACKAGE.
                                PERIOD_NUMBER_TYPE;
 WEEK_ARRAY              : in TIMETABLE_TYPES_PACKAGE.
                                WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD : in STAFF_FRAME_BASE_PACKAGE.
                                FRAME_BASE_RECORD_TYPE)
return BOOLEAN;

procedure MAKE_ALL_STAFF_BUSY
(STAFF_LIST              : in       SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING_LIST_PACKAGE.
                                    LIST_TYPE;
 MODULE                  : in       SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
 DAY                     : in       TIMETABLE_TYPES_PACKAGE.
                                    DAY_TYPE;
 PERIOD                  : in       TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE;
 WEEK_ARRAY              : in       TIMETABLE_TYPES_PACKAGE.
                                    WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD : in out STAFF_FRAME_BASE_PACKAGE.
                                    FRAME_BASE_RECORD_TYPE);

procedure MAKE_ALL_STAFF_FREE
(STAFF_LIST              : in       SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING_LIST_PACKAGE.
                                    LIST_TYPE;
 MODULE                  : in       SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
 DAY                     : in       TIMETABLE_TYPES_PACKAGE.
```

```
                                        DAY_TYPE;
   PERIOD                   : in        TIMETABLE_TYPES_PACKAGE.
                                        PERIOD_NUMBER_TYPE;
   WEEK_ARRAY               : in        TIMETABLE_TYPES_PACKAGE.
                                        WEEK_ARRAY_TYPE;
   STAFF_FRAME_BASE_RECORD  : in out    STAFF_FRAME_BASE_PACKAGE.
                                        FRAME_BASE_RECORD_TYPE);

  function ALL_STAFF_FREE
  (STAFF_LIST               : in        SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING_LIST_PACKAGE.
                                        LIST_TYPE;
   MODULE                   : in        SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.
                                        STRING;
   DAY                      : in        TIMETABLE_TYPES_PACKAGE.
                                        DAY_TYPE;
   PERIOD                   : in        TIMETABLE_TYPES_PACKAGE.
                                        PERIOD_NUMBER_TYPE;
   WEEK_ARRAY               : in        TIMETABLE_TYPES_PACKAGE.
                                        WEEK_ARRAY_TYPE;
   STAFF_FRAME_BASE_RECORD  : in        STAFF_FRAME_BASE_PACKAGE.
                                        FRAME_BASE_RECORD_TYPE)
  return BOOLEAN;

-------------------------------------------------------------------------
end STAFF_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------------
--
-- Unit      : STAFF_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 21 JULY 1992
-- Function  : This package provides the operations to build a
--             STAFF Frame System
--
-------------------------------------------------------------------------
package body STAFF_PACKAGE is
-------------------------------------------------------------------------

  use TIMETABLE_TYPES_PACKAGE;  -- For 'or' operation

  -----------------------------------------------------------------------
  procedure MAKE_BUSY
  -----------------------------------------------------------------------
  (LECTURER                 : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     STRING;
   MODULE                   : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     STRING;
   DAY                      : in     TIMETABLE_TYPES_PACKAGE.
                                     DAY_TYPE;
   PERIOD                   : in     TIMETABLE_TYPES_PACKAGE.
                                     PERIOD_NUMBER_TYPE;
   WEEK_ARRAY               : in     TIMETABLE_TYPES_PACKAGE.
                                     WEEK_ARRAY_TYPE;
   STAFF_FRAME_BASE_RECORD  : in out STAFF_FRAME_BASE_PACKAGE.
                                     FRAME_BASE_RECORD_TYPE) is
  -----------------------------------------------------------------------

    MODULE_AVAILABILITY_PTR  : TIMETABLE_TYPES_PACKAGE.
                               STAFF_FACET_RECORD_PTR_TYPE;

    TOTAL_AVAILABILITY_PTR   : TIMETABLE_TYPES_PACKAGE.
                               STAFF_FACET_RECORD_PTR_TYPE;

    TOTAL_AVAILABILITY       : SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;

    SUBJECTS                 : SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
  -----------------------------------------------------------------------
  begin

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_SUBSTRING => "ALL",
     TO_THE_STRING      => TOTAL_AVAILABILITY);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_SUBSTRING => "SUBJECTS",
```

```
    TO_THE_STRING      => SUBJECTS);

  STAFF_FRAME_BASE_PACKAGE.
  FIND_FACET
  (FACET_NAME         => TOTAL_AVAILABILITY,
   SLOT_NAME          => SUBJECTS,
   FRAME_NAME         => LECTURER,
   FRAME_BASE_RECORD  => STAFF_FRAME_BASE_RECORD,
   FACET_PTR          => TOTAL_AVAILABILITY_PTR);

  TOTAL_AVAILABILITY_PTR.
  AVAILABILITY_MATRIX(DAY, PERIOD) := TOTAL_AVAILABILITY_PTR.
                                      AVAILABILITY_MATRIX
                                      (DAY, PERIOD) or
                                      WEEK_ARRAY;

-------------------------------------------------------------------
end MAKE_BUSY;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure MAKE_FREE
-------------------------------------------------------------------
(LECTURER                 : in     SYSTEM_TYPES_PACKAGE.
                                   DYNAMIC_STRING.
                                   STRING;
 MODULE                   : in     SYSTEM_TYPES_PACKAGE.
                                   DYNAMIC_STRING.
                                   STRING;
 DAY                      : in     TIMETABLE_TYPES_PACKAGE.
                                   DAY_TYPE;
 PERIOD                   : in     TIMETABLE_TYPES_PACKAGE.
                                   PERIOD_NUMBER_TYPE;
 WEEK_ARRAY               : in     TIMETABLE_TYPES_PACKAGE.
                                   WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD  : in out STAFF_FRAME_BASE_PACKAGE.
                                   FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
  MODULE_AVAILABILITY_PTR  : TIMETABLE_TYPES_PACKAGE.
                             STAFF_FACET_RECORD_PTR_TYPE;

  TOTAL_AVAILABILITY_PTR   : TIMETABLE_TYPES_PACKAGE.
                             STAFF_FACET_RECORD_PTR_TYPE;

  TOTAL_AVAILABILITY       : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;

  SUBJECTS                 : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "ALL",
   TO_THE_STRING      => TOTAL_AVAILABILITY);
```

```
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_SUBSTRING => "SUBJECTS",
    TO_THE_STRING      => SUBJECTS);

   STAFF_FRAME_BASE_PACKAGE.
   FIND_FACET
   (FACET_NAME         => TOTAL_AVAILABILITY,
    SLOT_NAME          => SUBJECTS,
    FRAME_NAME         => LECTURER,
    FRAME_BASE_RECORD  => STAFF_FRAME_BASE_RECORD,
    FACET_PTR          => TOTAL_AVAILABILITY_PTR);

   TOTAL_AVAILABILITY_PTR.
   AVAILABILITY_MATRIX(DAY, PERIOD) := TOTAL_AVAILABILITY_PTR.
                                       AVAILABILITY_MATRIX
                                       (DAY, PERIOD) and
                                       (not WEEK_ARRAY);
-------------------------------------------------------------------
end MAKE_FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
function IS_FREE
-------------------------------------------------------------------
(LECTURER                 : in SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
 DAY                      : in TIMETABLE_TYPES_PACKAGE.
                              DAY_TYPE;
 PERIOD                   : in TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
 WEEK_ARRAY               : in TIMETABLE_TYPES_PACKAGE.
                              WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD  : in STAFF_FRAME_BASE_PACKAGE.
                              FRAME_BASE_RECORD_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
  TOTAL_AVAILABILITY_PTR   : TIMETABLE_TYPES_PACKAGE.
                             STAFF_FACET_RECORD_PTR_TYPE;

  TOTAL_AVAILABILITY       : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;

  SUBJECTS                 : SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "ALL",
   TO_THE_STRING      => TOTAL_AVAILABILITY);
```

```
   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_SUBSTRING => "SUBJECTS",
    TO_THE_STRING      => SUBJECTS);

   STAFF_FRAME_BASE_PACKAGE.
   FIND_FACET
   (FACET_NAME         => TOTAL_AVAILABILITY,
    SLOT_NAME          => SUBJECTS,
    FRAME_NAME         => LECTURER,
    FRAME_BASE_RECORD  => STAFF_FRAME_BASE_RECORD,
    FACET_PTR          => TOTAL_AVAILABILITY_PTR);

   return
   (TOTAL_AVAILABILITY_PTR.
    AVAILABILITY_MATRIX(DAY, PERIOD) and WEEK_ARRAY) =
    TIMETABLE_TYPES_PACKAGE.FREE;

-------------------------------------------------------------------
end IS_FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure MAKE_ALL_STAFF_BUSY
-------------------------------------------------------------------
(STAFF_LIST                 : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING_LIST_PACKAGE.
                                     LIST_TYPE;
 MODULE                     : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     STRING;
 DAY                        : in     TIMETABLE_TYPES_PACKAGE.
                                     DAY_TYPE;
 PERIOD                     : in     TIMETABLE_TYPES_PACKAGE.
                                     PERIOD_NUMBER_TYPE;
 WEEK_ARRAY                 : in     TIMETABLE_TYPES_PACKAGE.
                                     WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD    : in out STAFF_FRAME_BASE_PACKAGE.
                                     FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
begin

  for STAFF_MEMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LENGTH_OF
                           (LIST => STAFF_LIST)
  loop
    MAKE_BUSY(
      LECTURER => SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING_LIST_PACKAGE.
                  ITEM_AT(
                  LIST        => STAFF_LIST,
                  NODE_NUMBER => STAFF_MEMBER),
      MODULE                  => MODULE,
      DAY                     => DAY,
      PERIOD                  => PERIOD,
      WEEK_ARRAY              => WEEK_ARRAY,
```

```
      STAFF_FRAME_BASE_RECORD => STAFF_FRAME_BASE_RECORD);
  end loop;

-------------------------------------------------------------------
end MAKE_ALL_STAFF_BUSY;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure MAKE_ALL_STAFF_FREE
-------------------------------------------------------------------
(STAFF_LIST                : in     SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING_LIST_PACKAGE.
                                    LIST_TYPE;
 MODULE                    : in     SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
 DAY                       : in     TIMETABLE_TYPES_PACKAGE.
                                    DAY_TYPE;
 PERIOD                    : in     TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE;
 WEEK_ARRAY                : in     TIMETABLE_TYPES_PACKAGE.
                                    WEEK_ARRAY_TYPE;
 STAFF_FRAME_BASE_RECORD   : in out STAFF_FRAME_BASE_PACKAGE.
                                    FRAME_BASE_RECORD_TYPE) is
-------------------------------------------------------------------
begin

  for STAFF_MEMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LENGTH_OF
                           (LIST => STAFF_LIST)
  loop
    MAKE_FREE(
      LECTURER => SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING_LIST_PACKAGE.
                  ITEM_AT(
                  LIST        => STAFF_LIST,
                  NODE_NUMBER => STAFF_MEMBER),
      MODULE                  => MODULE,
      DAY                     => DAY,
      PERIOD                  => PERIOD,
      WEEK_ARRAY              => WEEK_ARRAY,
      STAFF_FRAME_BASE_RECORD => STAFF_FRAME_BASE_RECORD);
  end loop;

-------------------------------------------------------------------
end MAKE_ALL_STAFF_FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
function ALL_STAFF_FREE
-------------------------------------------------------------------
(STAFF_LIST                : in     SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING_LIST_PACKAGE.
                                    LIST_TYPE;
 MODULE                    : in     SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    STRING;
```

```
   DAY                        : in     TIMETABLE_TYPES_PACKAGE.
                                       DAY_TYPE;
   PERIOD                     : in     TIMETABLE_TYPES_PACKAGE.
                                       PERIOD_NUMBER_TYPE;
   WEEK_ARRAY                 : in     TIMETABLE_TYPES_PACKAGE.
                                       WEEK_ARRAY_TYPE;
   STAFF_FRAME_BASE_RECORD    : in     STAFF_FRAME_BASE_PACKAGE.
                                       FRAME_BASE_RECORD_TYPE)
  return BOOLEAN is
  ------------------------------------------------------------------
  begin

    for STAFF_MEMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_LIST_PACKAGE.
                             LENGTH_OF
                             (LIST => STAFF_LIST)
    loop
      if not IS_FREE(
             LECTURER => SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING_LIST_PACKAGE.
                         ITEM_AT(
                         LIST        => STAFF_LIST,
                         NODE_NUMBER => STAFF_MEMBER),
             DAY                     => DAY,
             PERIOD                  => PERIOD,
             WEEK_ARRAY              => WEEK_ARRAY,
             STAFF_FRAME_BASE_RECORD => STAFF_FRAME_BASE_RECORD)
      then

        return FALSE;
      end if;
    end loop;

    return TRUE;

  ------------------------------------------------------------------
  end ALL_STAFF_FREE;
  ------------------------------------------------------------------

begin

  STAFF_FRAME_BASE_PACKAGE.
  BUILD
  (FRAME_BASE_RECORD        => STAFF_FRAME_BASE_RECORD,
   FRAME_FILE_LIST_FILENAME => "staffschoolfile.list");

------------------------------------------------------------------
end STAFF_PACKAGE;
------------------------------------------------------------------
```

Module

```
------------------------------------------------------------------------
--
-- Unit      : MODULE_PACKAGE specification
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 24 JULY 1992
-- Function  : This package provides the operations to build a
--             Module Frame System
--
------------------------------------------------------------------------
with GENERIC_FRAME_BASE_PACKAGE,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TEXT_IO;
------------------------------------------------------------------------
package MODULE_PACKAGE is
------------------------------------------------------------------------

  -- Integrate a frame abstract knowledge type

  package MODULE_FRAME_BASE_PACKAGE is new
          GENERIC_FRAME_BASE_PACKAGE
  (SLOT_TYPE              => TIMETABLE_TYPES_PACKAGE.
                             MODULE_SLOT_TYPE,
   FACET_RECORD_TYPE      => TIMETABLE_TYPES_PACKAGE.
                             MODULE_FACET_RECORD_TYPE,
   FACET_RECORD_PTR_TYPE  => TIMETABLE_TYPES_PACKAGE.
                             MODULE_FACET_RECORD_PTR_TYPE,
   "<"                    => TIMETABLE_TYPES_PACKAGE.
                             "<",
   ">"                    => TIMETABLE_TYPES_PACKAGE.
                             ">",
   IS_EQUAL               => TIMETABLE_TYPES_PACKAGE.
                             IS_EQUAL,
   IS_LESS_THAN           => TIMETABLE_TYPES_PACKAGE.
                             IS_LESS_THAN,
   IS_GREATER_THAN        => TIMETABLE_TYPES_PACKAGE.
                             IS_GREATER_THAN,
   PUT                    => TIMETABLE_TYPES_PACKAGE.
                             PUT,
   GET                    => TIMETABLE_TYPES_PACKAGE.
                             GET);

  MODULE_FRAME_BASE_RECORD : MODULE_FRAME_BASE_PACKAGE.
                              FRAME_BASE_RECORD_TYPE;

  -- Define additional user application operations

  procedure FIND_STAFF
  (MODULE                   : in     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING.
                                     STRING;
   STAFF_SLOT_PTR           : in out MODULE_FRAME_BASE_PACKAGE.
                                     SLOT_PACKAGE.
                                     SLOT_RECORD_PTR_TYPE;
   MODULE_FRAME_BASE_RECORD : in     MODULE_FRAME_BASE_PACKAGE.
                                     FRAME_BASE_RECORD_TYPE);
```

```
-------------------------------------------------------------------------
end MODULE_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : MODULE_PACKAGE body
-- Author    : A Harrison Software Engineering Group,
--             Cranfield University, RMCS, Shrivenam
-- Date      : 24 JULY 1992
-- Function  : This package provides the operations to build a
--             Module Frame System
--
-------------------------------------------------------------------
package body MODULE_PACKAGE is
-------------------------------------------------------------------

  -----------------------------------------------------------------
  procedure FIND_STAFF
  -----------------------------------------------------------------
  (MODULE                     : in     SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       STRING;
   STAFF_SLOT_PTR             : in out MODULE_FRAME_BASE_PACKAGE.
                                       SLOT_PACKAGE.
                                       SLOT_RECORD_PTR_TYPE;
   MODULE_FRAME_BASE_RECORD   : in     MODULE_FRAME_BASE_PACKAGE.
                                       FRAME_BASE_RECORD_TYPE) is
  -----------------------------------------------------------------
    STAFF_SLOT_NAME : SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      STRING;
  -----------------------------------------------------------------
  begin

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_SUBSTRING => "Staff",
     TO_THE_STRING      => STAFF_SLOT_NAME);

    MODULE_FRAME_BASE_PACKAGE.
    FIND_SLOT
    (SLOT_NAME          => STAFF_SLOT_NAME,
     FRAME_NAME         => MODULE,
     FRAME_BASE_RECORD  => MODULE_FRAME_BASE_RECORD,
     SLOT_PTR           => STAFF_SLOT_PTR);

  -----------------------------------------------------------------
  end FIND_STAFF;
  -----------------------------------------------------------------

begin

  MODULE_FRAME_BASE_PACKAGE.
  BUILD
  (FRAME_BASE_RECORD         => MODULE_FRAME_BASE_RECORD,
   FRAME_FILE_LIST_FILENAME => "staffmodulefile.list");

-------------------------------------------------------------------
end MODULE_PACKAGE;
-------------------------------------------------------------------
```

Staff KS

```
------------------------------------------------------------------------
--
-- Unit      : STAFF_KS_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 27 July 1992
-- Function  : This package provides the blackboard transformation on
--             the degree_activities level by alllocating staff
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------------
package STAFF_KS_PACKAGE is
------------------------------------------------------------------------

  procedure PROCESS_EVENT
  (ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD   : in out TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_TYPE);

------------------------------------------------------------------------
end STAFF_KS_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : STAFF_KS_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 27 July 1992
-- Function  : This package provides the blackboard transformation on
--             the degree_activity level by allocating staff
--
-------------------------------------------------------------------
package body STAFF_KS_PACKAGE is
-------------------------------------------------------------------

  -----------------------------------------------------------------
  --
  -- Process_event finds the appropriate staff member and updates
  -- the activity
  --
  -----------------------------------------------------------------
  procedure PROCESS_EVENT
  -----------------------------------------------------------------
  (ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD   : in out TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_TYPE) is
  -----------------------------------------------------------------

    STAFF_SLOT_PTR : MODULE_PACKAGE.
                     MODULE_FRAME_BASE_PACKAGE.
                     SLOT_PACKAGE.
                     SLOT_RECORD_PTR_TYPE;

  -----------------------------------------------------------------
  begin

    -- Find the staff

    MODULE_PACKAGE.
    FIND_STAFF
    (MODULE                   => ACTIVITY_PTR.
                                 FACT,
     STAFF_SLOT_PTR           => STAFF_SLOT_PTR,
     MODULE_FRAME_BASE_RECORD => MODULE_PACKAGE.
                                 MODULE_FRAME_BASE_RECORD);

    -- Convert staff tree to an array

    declare

      STAFF_ARRAY : MODULE_PACKAGE.
                    MODULE_FRAME_BASE_PACKAGE.
                    FACET_PACKAGE.
                    FACET_TREE_PACKAGE.
                    ITEM_ARRAY_TYPE
                    (1 .. MODULE_PACKAGE.
                          MODULE_FRAME_BASE_PACKAGE.
                          FACET_PACKAGE.
                          FACET_TREE_PACKAGE.
```

```
                    SIZE_OF
                    (TREE_PTR => STAFF_SLOT_PTR.
                                 FACET_TREE_PTR));

  ITEM_NUMBER : NATURAL := 0;

begin

  MODULE_PACKAGE.
  MODULE_FRAME_BASE_PACKAGE.
  FACET_PACKAGE.
  FACET_TREE_PACKAGE.
  FORM_ARRAY
  (TREE_PTR    => STAFF_SLOT_PTR.
                  FACET_TREE_PTR,
   ITEM_ARRAY  => STAFF_ARRAY,
   ITEM_NUMBER => ITEM_NUMBER);

  -- Copy contents of staff array to staff list in activity node

  for I in 1 .. ITEM_NUMBER
  loop

    declare
      TEMP_STRING : SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    STRING;
    begin
      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_STRING => STAFF_ARRAY(I).
                          INITIALS,
       TO_THE_STRING   => TEMP_STRING);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => ACTIVITY_PTR.
                    STAFF_LIST,
       THIS_ITEM => TEMP_STRING);

      -- Add staff to fused data

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      APPEND
      (THE_SUBSTRING => "_",
       TO_THE_STRING => ACTIVITY_PTR.
                        FUSED_FACT);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      APPEND
      (THE_STRING    => TEMP_STRING,
       TO_THE_STRING => ACTIVITY_PTR.
                        FUSED_FACT);
    end;

  end loop;
```

```
    end;

    -- Change action to "Select Period"

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_SUBSTRING => "Select Period",
     TO_THE_STRING      => ACTIVITY_PTR.
                           ACTION);

    -- Add activity to the event list

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   EVENT_LISTS,
                    ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                   PERIOD_EVENTS).
                                   LIST,
      THIS_ITEM => ACTIVITY_PTR);

  ------------------------------------------------------------------
  exception
    when others =>
      TEXT_IO.PUT_LINE("Exception raised in STAFF_KS_PACKAGE" &
                       " in PROCESS_EVENT");
  ------------------------------------------------------------------
  end PROCESS_EVENT;
  ------------------------------------------------------------------

------------------------------------------------------------------
end STAFF_KS_PACKAGE;
------------------------------------------------------------------
```

Period_1

```
------------------------------------------------------------------------
--
-- Unit      : PERIOD_1_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield Univerdsity, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides period utilities
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE;
------------------------------------------------------------------------
package PERIOD_1_PACKAGE is
------------------------------------------------------------------------

  -- Integrate two logic abstract knowledge types

  package ESE_PERIOD_INFERENCE_PACKAGE is new
          LOGIC_INFERENCE_PACKAGE("ESE");
  package IT_PERIOD_INFERENCE_PACKAGE is new
          LOGIC_INFERENCE_PACKAGE("IT");

  package DAY_IO is new TEXT_IO.
                        ENUMERATION_IO
                        (TIMETABLE_TYPES_PACKAGE.
                         DAY_TYPE);

  package PERIOD_IO is new TEXT_IO.
                           INTEGER_IO
                           (TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE);

  type PERIOD_DETAILS_TYPE is
  (DAY, FIRST_PERIOD, SECOND_PERIOD, THIRD_PERIOD,
   P1W1, P1W2, P1W3, P1W4, P1W5, P1W6, P1W7, P1W8, P1W9,
   P1W10,P1W11, P2W1, P2W2, P2W3, P2W4, P2W5, P2W6, P2W7, P2W8,
   P2W9, P2W10,P2W11, P3W1, P3W2, P3W3, P3W4, P3W5, P3W6, P3W7,
   P3W8, P3W9, P3W10, P3W11);

  ESE_PERIOD_KB : ESE_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  KB_RECORD;
  IT_PERIOD_KB  : IT_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  KB_RECORD;

------------------------------------------------------------------------
end PERIOD_1_PACKAGE;
------------------------------------------------------------------------
```

Period 2

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_2_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides period utilities
--
-------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE;
use  PERIOD_1_PACKAGE;
-------------------------------------------------------------------
package PERIOD_2_PACKAGE is
-------------------------------------------------------------------
  procedure PRINT;

  procedure STRIP_SPACES
  (IN_STANDARD_STRING : in     STANDARD.
                               STRING;
   OUT_DYNAMIC_STRING : in out SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING);

  procedure GET_WEEK
  (INDEX       :    out NATURAL;
   WEEK_CODE   : in     STANDARD.
                        STRING;
   POSITION    : in out NATURAL;
   TOKEN       :    out CHARACTER);

  procedure CONVERT
  (WEEK_CODE   : in     STANDARD.
                        STRING;
   WEEK_ARRAY  :    out TIMETABLE_TYPES_PACKAGE.
                        WEEK_ARRAY_TYPE);

  procedure EXTRACT_WEEKS
  (WEEKS_REQUIRED : in     TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
   OLD_CLAUSE     : in out STANDARD.
                           STRING;
   NEW_CLAUSE     : in out STANDARD.
                           STRING;
   VALUE_LIST     : in out SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LIST_TYPE);
-------------------------------------------------------------------
end PERIOD_2_PACKAGE;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_2_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides the period utilities
--
-------------------------------------------------------------------
package body PERIOD_2_PACKAGE is
-------------------------------------------------------------------

  -----------------------------------------------------------------
  procedure PRINT is
  -----------------------------------------------------------------
  begin

    ESE_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    PRINT
    (KB => ESE_PERIOD_KB);

    TEXT_IO.NEW_LINE(2);

    IT_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    PRINT
    (KB => IT_PERIOD_KB);

    TEXT_IO.NEW_LINE(2);

  -----------------------------------------------------------------
  end PRINT;
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  procedure STRIP_SPACES
  -----------------------------------------------------------------
  (IN_STANDARD_STRING : in     STANDARD.
                               STRING;
   OUT_DYNAMIC_STRING : in out SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING) is
  -----------------------------------------------------------------
  begin
    for CHAR in 1 .. IN_STANDARD_STRING'LENGTH
    loop
      if IN_STANDARD_STRING(CHAR) /= ' ' then

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        APPEND
        (THE_ITEM => IN_STANDARD_STRING(CHAR),
         TO_THE_STRING => OUT_DYNAMIC_STRING);

      end if;
    end loop;
  -----------------------------------------------------------------
  end STRIP_SPACES;
```

```
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Get_week converts the week code into an appropriate index
--
-------------------------------------------------------------------
procedure GET_WEEK
-------------------------------------------------------------------
(INDEX     :    out NATURAL;
 WEEK_CODE : in     STANDARD.
                    STRING;
 POSITION  : in out NATURAL;
 TOKEN     :    out CHARACTER) is
-------------------------------------------------------------------
  FIRST : NATURAL := POSITION;
-------------------------------------------------------------------
begin

  -- Get the week code

  while WEEK_CODE(POSITION) /= ',' and then
        WEEK_CODE(POSITION) /= '-'
  loop
    POSITION := POSITION + 1;
    exit when POSITION > WEEK_CODE'LENGTH;
  end loop;

  -- Transform it into an index

  INDEX := NATURAL'VALUE(WEEK_CODE(FIRST .. POSITION - 1));

  -- Pass back the next token

  if POSITION > WEEK_CODE'LENGTH then
    TOKEN := ',';
  else
    TOKEN := WEEK_CODE(POSITION);
    POSITION := POSITION + 1;
  end if;

exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE " &
     "at GET_WEEK");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_KS_PACKAGE " &
     "at GET_WEEK");
-------------------------------------------------------------------
end GET_WEEK;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Convert transforms the week code into a boolean array. For
-- example
```

```
-- 1,3,5-11 is transformed into TFTFTTTTTTT
--
-------------------------------------------------------------------
procedure CONVERT
-------------------------------------------------------------------
(WEEK_CODE  : in      STANDARD.
                      STRING;
 WEEK_ARRAY :     out TIMETABLE_TYPES_PACKAGE.
                      WEEK_ARRAY_TYPE) is
-------------------------------------------------------------------
  TOKEN       : CHARACTER;
  POSITION    : NATURAL := 1;
  FIRST_INDEX : NATURAL;
  LAST_INDEX  : NATURAL;
-------------------------------------------------------------------
begin

  WEEK_ARRAY := (1 .. WEEK_ARRAY'LENGTH => FALSE);

  while POSITION <= WEEK_CODE'LENGTH
  loop

    -- Get the start week

    GET_WEEK
    (INDEX     => FIRST_INDEX,
     WEEK_CODE => WEEK_CODE,
     POSITION  => POSITION,
     TOKEN     => TOKEN);

    if TOKEN = ',' then

      -- A single period

      WEEK_ARRAY(FIRST_INDEX) := TRUE;

    else

      -- Multiple periods get the final week

      GET_WEEK
      (INDEX     => LAST_INDEX,
       WEEK_CODE => WEEK_CODE,
       POSITION  => POSITION,
       TOKEN     => TOKEN);

      WEEK_ARRAY(FIRST_INDEX .. LAST_INDEX) :=
                (FIRST_INDEX .. LAST_INDEX   => TRUE);

    end if;

  end loop;
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at CONVERT");
-------------------------------------------------------------------
end CONVERT;
-------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Extract weeks selects the current weeks and forms the old and
-- new periods
--
------------------------------------------------------------------
procedure EXTRACT_WEEKS
------------------------------------------------------------------
(WEEKS_REQUIRED : in     TIMETABLE_TYPES_PACKAGE.
                         WEEK_ARRAY_TYPE;
 OLD_CLAUSE     : in out STANDARD.
                         STRING;
 NEW_CLAUSE     : in out STANDARD.
                         STRING;
 VALUE_LIST     : in out SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING_LIST_PACKAGE.
                         LIST_TYPE) is
------------------------------------------------------------------

  CONSTRAINT_ERROR : exception;

------------------------------------------------------------------
begin

  -- List format -> A o B x C o D x E x .. K o

  for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          LENGTH_OF
                          (LIST => VALUE_LIST) / 2
                           -- two at a time
  loop

  declare

    ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;
    VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;
  begin

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => ITEM_DYNAMIC_STRING);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => VALUE_DYNAMIC_STRING);

    declare

      ITEM  : STANDARD.
              CHARACTER := SYSTEM_TYPES_PACKAGE.
```

```
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ITEM_DYNAMIC_STRING)(1);

  ITEM_VALUE : STANDARD.
               STRING(1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           LENGTH_OF
                           (THE_STRING =>
                            VALUE_DYNAMIC_STRING))) :=
                            SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            SUBSTRING_OF
                            (THE_STRING =>
                             VALUE_DYNAMIC_STRING);
begin

  case ITEM is

    when 'A' => OLD_CLAUSE(19) := ITEM_VALUE(1);

                NEW_CLAUSE(19) := ITEM_VALUE(1);

    when 'B' => OLD_CLAUSE(21) := ITEM_VALUE(1);

                NEW_CLAUSE(21) := ITEM_VALUE(1);

    when 'C' => OLD_CLAUSE(23) := ITEM_VALUE(1);

                NEW_CLAUSE(23) := ITEM_VALUE(1);

    when 'D' => OLD_CLAUSE(25) := ITEM_VALUE(1);

                NEW_CLAUSE(25) := ITEM_VALUE(1);

    when 'E' => OLD_CLAUSE(27) := ITEM_VALUE(1);

                NEW_CLAUSE(27) := ITEM_VALUE(1);

    when 'F' => OLD_CLAUSE(29) := ITEM_VALUE(1);

                NEW_CLAUSE(29) := ITEM_VALUE(1);

    when 'G' => OLD_CLAUSE(31) := ITEM_VALUE(1);

                NEW_CLAUSE(31) := ITEM_VALUE(1);

    when 'H' => OLD_CLAUSE(33) := ITEM_VALUE(1);

                NEW_CLAUSE(33) := ITEM_VALUE(1);

    when 'I' => OLD_CLAUSE(35) := ITEM_VALUE(1);

                NEW_CLAUSE(35) := ITEM_VALUE(1);

    when 'J' => OLD_CLAUSE(37) := ITEM_VALUE(1);

                NEW_CLAUSE(37) := ITEM_VALUE(1);

    when 'K' => OLD_CLAUSE(39) := ITEM_VALUE(1);
```

```
                        NEW_CLAUSE(39) := ITEM_VALUE(1);

          when others => raise CONSTRAINT_ERROR;
        end case;

      end;

    end;

    end loop;

    -- Update new clause to weeks required

    declare
      OFFSET : POSITIVE := 18;
    begin
      for WEEK in 1 .. TIMETABLE_TYPES_PACKAGE.
                       NUMBER_OF_WEEKS
      loop

       if WEEKS_REQUIRED(WEEK) = TRUE then
          NEW_CLAUSE(WEEK + OFFSET) := 'x';
       end if;
       OFFSET := OFFSET + 1;
     end loop;
    end;

  exception
    when CONSTRAINT_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE at " &
       "EXTRACT_WEEKS");
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_KS_PACKAGE " &
       "at EXTRACT_WEEKS");
  ------------------------------------------------------------------
  end EXTRACT_WEEKS;
  ------------------------------------------------------------------

------------------------------------------------------------------
end PERIOD_2_PACKAGE;
------------------------------------------------------------------
```

Period 3

```
------------------------------------------------------------------------
--
-- Unit      : PERIOD_3_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function : This package provides period utilities
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE;
use  PERIOD_1_PACKAGE;
------------------------------------------------------------------------
package PERIOD_3_PACKAGE is
------------------------------------------------------------------------

  procedure EXTRACT_ONE_PERIOD
  (PERIOD         :     out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
   DAY            :     out TIMETABLE_TYPES_PACKAGE.
                            DAY_TYPE;
   OLD_CLAUSE     : in out STANDARD.
                            STRING;
   NEW_CLAUSE     : in out STANDARD.
                            STRING;
   VALUE_LIST     : in out SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE);

  procedure EXTRACT_TWO_PERIODS
  (PERIOD_1       :     out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
   PERIOD_2       :     out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
   THIS_DAY       :     out TIMETABLE_TYPES_PACKAGE.
                            DAY_TYPE;
   OLD_P1_CLAUSE  : in out STANDARD.
                            STRING;
   OLD_P2_CLAUSE  : in out STANDARD.
                            STRING;
   NEW_P1_CLAUSE  : in out STANDARD.
                            STRING;
   NEW_P2_CLAUSE  : in out STANDARD.
                            STRING;
   VALUE_LIST     : in out SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE);

  procedure EXTRACT_THREE_PERIODS
  (PERIOD_1       :     out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
```

```
 PERIOD_2          :    out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
 PERIOD_3          :    out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
 THIS_DAY          :    out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
 OLD_P1_CLAUSE     : in out STANDARD.
                           STRING;
 OLD_P2_CLAUSE     : in out STANDARD.
                           STRING;
 OLD_P3_CLAUSE     : in out STANDARD.
                           STRING;
 NEW_P1_CLAUSE     : in out STANDARD.
                           STRING;
 NEW_P2_CLAUSE     : in out STANDARD.
                           STRING;
 NEW_P3_CLAUSE     : in out STANDARD.
                           STRING;
 VALUE_LIST        : in out SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LIST_TYPE);

function COMMON_MODULES_AVAILABLE
(PERIOD_QUERY_STRING : in STANDARD.
                          STRING;
 ACTIVITY_PTR        : in TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_NODE_PTR_TYPE) return BOOLEAN;

procedure ADJUST_COMMON_MODULE_PERIODS
(DAY              : in STANDARD.
                       STRING;
 PERIOD           : in STANDARD.
                       STRING;
 WEEKS_REQUIRED   : in TIMETABLE_TYPES_PACKAGE.
                       WEEK_ARRAY_TYPE;
 ACTIVITY_PTR     : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE);

procedure ADD_PERIOD_DETAILS
(THIS_DAY         : in      TIMETABLE_TYPES_PACKAGE.
                            DAY_TYPE;
 FIRST_PERIOD     : in      TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
 LAST_PERIOD      : in      TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
 WEEKS_REQUIRED   : in      TIMETABLE_TYPES_PACKAGE.
                            WEEK_ARRAY_TYPE;
 ACTIVITY_PTR     : in      TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD       : in out  TIMETABLE_BLACKBOARD_PACKAGE.
                            TIMETABLE_BLACKBOARD.
                            BLACKBOARD_TYPE);

function DEGREE
(ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
  NODE        : in NATURAL) return CHARACTER;

function CONVERT
(DAY : in TIMETABLE_TYPES_PACKAGE.
```

```
            DAY_TYPE) return TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;

  function CONVERT
  (DAY : in TIMETABLE_TYPES_PACKAGE.
            TIMETABLE_ITEM_TYPE) return TIMETABLE_TYPES_PACKAGE.
                                        DAY_TYPE;

-------------------------------------------------------------------------
end PERIOD_3_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_3_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides the period utilities
--
-------------------------------------------------------------------
with ROOM_PACKAGE;
-------------------------------------------------------------------
package body PERIOD_3_PACKAGE is
-------------------------------------------------------------------

  procedure PRINT is
  begin

    ESE_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    PRINT
    (KB => ESE_PERIOD_KB);

    TEXT_IO.NEW_LINE(2);

    IT_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    PRINT
    (KB => IT_PERIOD_KB);

    TEXT_IO.NEW_LINE(2);
  -------------------------------------------------------------------
  end PRINT;
  -------------------------------------------------------------------


  -------------------------------------------------------------------
  procedure STRIP_SPACES
  -------------------------------------------------------------------
  (IN_STANDARD_STRING : in     STANDARD.
                               STRING;
   OUT_DYNAMIC_STRING : in out SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING) is
  begin
  -------------------------------------------------------------------
    for CHAR in 1 .. IN_STANDARD_STRING'LENGTH
    loop
      if IN_STANDARD_STRING(CHAR) /= ' ' then

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        APPEND
        (THE_ITEM => IN_STANDARD_STRING(CHAR),
         TO_THE_STRING => OUT_DYNAMIC_STRING);

      end if;
    end loop;
  -------------------------------------------------------------------
  end STRIP_SPACES;
  -------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Get_week converts the week code into an appropriate index
--
-------------------------------------------------------------------
procedure GET_WEEK
-------------------------------------------------------------------
(INDEX     :    out NATURAL;
 WEEK_CODE : in     STANDARD.
                    STRING;
 POSITION  : in out NATURAL;
 TOKEN     :    out CHARACTER) is
-------------------------------------------------------------------
  FIRST : NATURAL := POSITION;
-------------------------------------------------------------------
begin

  -- Get the week code

  while WEEK_CODE(POSITION) /= ',' and then
        WEEK_CODE(POSITION) /= '-'
  loop
    POSITION := POSITION + 1;
    exit when POSITION > WEEK_CODE'LENGTH;
  end loop;

  -- Transform it into an index

  INDEX := NATURAL'VALUE(WEEK_CODE(FIRST .. POSITION - 1));

  -- Pass back the next token

  if POSITION > WEEK_CODE'LENGTH then
    TOKEN := ',';
  else
    TOKEN := WEEK_CODE(POSITION);
    POSITION := POSITION + 1;
  end if;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE " &
     "at GET_WEEK");
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at GET_WEEK");
-------------------------------------------------------------------
end GET_WEEK;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Convert transforms the week code into a boolean array. For
-- example
-- 1,3,5-11 is transformed into TFTFTTTTTTT
--
```

```
-------------------------------------------------------------------
procedure CONVERT
-------------------------------------------------------------------
(WEEK_CODE  : in     STANDARD.
                     STRING;
 WEEK_ARRAY :    out TIMETABLE_TYPES_PACKAGE.
                     WEEK_ARRAY_TYPE) is
-------------------------------------------------------------------
  TOKEN       : CHARACTER;
  POSITION    : NATURAL := 1;
  FIRST_INDEX : NATURAL;
  LAST_INDEX  : NATURAL;
-------------------------------------------------------------------
begin

  WEEK_ARRAY := (1 .. WEEK_ARRAY'LENGTH => FALSE);

  while POSITION <= WEEK_CODE'LENGTH
  loop

    -- Get the start week

    GET_WEEK
    (INDEX     => FIRST_INDEX,
     WEEK_CODE => WEEK_CODE,
     POSITION  => POSITION,
     TOKEN     => TOKEN);

    if TOKEN = ',' then

      -- A single period

      WEEK_ARRAY(FIRST_INDEX) := TRUE;

    else

      -- Multiple periods get the final week

      GET_WEEK
      (INDEX     => LAST_INDEX,
       WEEK_CODE => WEEK_CODE,
       POSITION  => POSITION,
       TOKEN     => TOKEN);

      WEEK_ARRAY(FIRST_INDEX .. LAST_INDEX) :=
                 (FIRST_INDEX .. LAST_INDEX   => TRUE);

    end if;

  end loop;
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at CONVERT");
-------------------------------------------------------------------
end CONVERT;
-------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Extract weeks selects the current weeks and forms the old and
-- new periods
--
------------------------------------------------------------------
procedure EXTRACT_WEEKS
------------------------------------------------------------------
(WEEKS_REQUIRED : in     TIMETABLE_TYPES_PACKAGE.
                         WEEK_ARRAY_TYPE;
 OLD_CLAUSE     : in out STANDARD.
                         STRING;
 NEW_CLAUSE     : in out STANDARD.
                         STRING;
 VALUE_LIST     : in out SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING_LIST_PACKAGE.
                         LIST_TYPE) is
------------------------------------------------------------------
  CONSTRAINT_ERROR : exception;
------------------------------------------------------------------
begin

  -- List format -> A o B x C o D x E x .. K o

  for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          LENGTH_OF
                          (LIST => VALUE_LIST) / 2
                          -- two at a time
  loop

  declare

    ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;
    VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;

  begin


    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => ITEM_DYNAMIC_STRING);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => VALUE_DYNAMIC_STRING);

    declare

      ITEM  : STANDARD.
              CHARACTER := SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
```

```
                        SUBSTRING_OF
                        (THE_STRING => ITEM_DYNAMIC_STRING)(1);

    ITEM_VALUE : STANDARD.
                 STRING(1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             LENGTH_OF
                             (THE_STRING =>
                              VALUE_DYNAMIC_STRING))) :=
                              SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              SUBSTRING_OF
                              (THE_STRING =>
                               VALUE_DYNAMIC_STRING);
  begin

    case ITEM is

      when 'A' => OLD_CLAUSE(19) := ITEM_VALUE(1);

                  NEW_CLAUSE(19) := ITEM_VALUE(1);

      when 'B' => OLD_CLAUSE(21) := ITEM_VALUE(1);

                  NEW_CLAUSE(21) := ITEM_VALUE(1);

      when 'C' => OLD_CLAUSE(23) := ITEM_VALUE(1);

                  NEW_CLAUSE(23) := ITEM_VALUE(1);

      when 'D' => OLD_CLAUSE(25) := ITEM_VALUE(1);

                  NEW_CLAUSE(25) := ITEM_VALUE(1);

      when 'E' => OLD_CLAUSE(27) := ITEM_VALUE(1);

                  NEW_CLAUSE(27) := ITEM_VALUE(1);

      when 'F' => OLD_CLAUSE(29) := ITEM_VALUE(1);

                  NEW_CLAUSE(29) := ITEM_VALUE(1);

      when 'G' => OLD_CLAUSE(31) := ITEM_VALUE(1);

                  NEW_CLAUSE(31) := ITEM_VALUE(1);

      when 'H' => OLD_CLAUSE(33) := ITEM_VALUE(1);

                  NEW_CLAUSE(33) := ITEM_VALUE(1);

      when 'I' => OLD_CLAUSE(35) := ITEM_VALUE(1);

                  NEW_CLAUSE(35) := ITEM_VALUE(1);

      when 'J' => OLD_CLAUSE(37) := ITEM_VALUE(1);

                  NEW_CLAUSE(37) := ITEM_VALUE(1);

      when 'K' => OLD_CLAUSE(39) := ITEM_VALUE(1);
```

```
                    NEW_CLAUSE(39) := ITEM_VALUE(1);

        when others => raise CONSTRAINT_ERROR;
       end case;

    end;

  end;

  end loop;

  -- Update new clause to weeks required

  declare
    OFFSET : POSITIVE := 18;
  begin
    for WEEK in 1 .. TIMETABLE_TYPES_PACKAGE.
                     NUMBER_OF_WEEKS
    loop

     if  WEEKS_REQUIRED(WEEK) = TRUE then
        NEW_CLAUSE(WEEK + OFFSET) := 'x';
     end if;
     OFFSET := OFFSET + 1;
   end loop;
  end;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE at " &
     "EXTRACT_WEEKS");
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at EXTRACT_WEEKS");
-------------------------------------------------------------------
end EXTRACT_WEEKS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Extract one period gets the selected period and day from the
-- result of a query
--
-------------------------------------------------------------------
procedure EXTRACT_ONE_PERIOD
-------------------------------------------------------------------
(PERIOD         :     out TIMETABLE_TYPES_PACKAGE.
                          PERIOD_NUMBER_TYPE;
 DAY            :     out TIMETABLE_TYPES_PACKAGE.
                          DAY_TYPE;
 OLD_CLAUSE     : in out STANDARD.
                          STRING;
 NEW_CLAUSE     : in out STANDARD.
                          STRING;
 VALUE_LIST     : in out SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          LIST_TYPE) is
-------------------------------------------------------------------
```

```
  ITEM_DYNAMIC_STRING   : SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          STRING;
  VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          STRING;

  CONSTRAINT_ERROR : exception;

-------------------------------------------------------------------
begin

  -- List format -> A  m o n  B  p 3 C 0 D x E x .. M o which
  -- formed query

  for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          LENGTH_OF
                          (LIST => VALUE_LIST) / 2
                           -- two at a time
  loop

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => ITEM_DYNAMIC_STRING);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => VALUE_DYNAMIC_STRING);

    declare

      ITEM   : STANDARD.
               CHARACTER := SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            SUBSTRING_OF
                            (THE_STRING => ITEM_DYNAMIC_STRING)(1);

      ITEM_VALUE : STANDARD.
                   STRING(1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.
                                        LENGTH_OF
                                        (THE_STRING =>
                                         VALUE_DYNAMIC_STRING))) :=
                                         SYSTEM_TYPES_PACKAGE.
                                         DYNAMIC_STRING.
                                         SUBSTRING_OF
                                         (THE_STRING =>
                                          VALUE_DYNAMIC_STRING);
    begin

      case ITEM is

        when 'A' => DAY := TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE'VALUE
```

```
                        (ITEM_VALUE);

                OLD_CLAUSE(8 .. 10) := ITEM_VALUE;
                NEW_CLAUSE(8 .. 10) := ITEM_VALUE;

    when 'B' => PERIOD := TIMETABLE_TYPES_PACKAGE.
                          PERIOD_NUMBER_TYPE'VALUE
                          (ITEM_VALUE(2 .. 2));

                OLD_CLAUSE(12 .. 13) := ITEM_VALUE;
                NEW_CLAUSE(12 .. 13) := ITEM_VALUE;

    when 'C' => OLD_CLAUSE(19) := ITEM_VALUE(1);

                NEW_CLAUSE(19) := ITEM_VALUE(1);

    when 'D' => OLD_CLAUSE(21) := ITEM_VALUE(1);

                NEW_CLAUSE(21) := ITEM_VALUE(1);

    when 'E' => OLD_CLAUSE(23) := ITEM_VALUE(1);

                NEW_CLAUSE(23) := ITEM_VALUE(1);

    when 'F' => OLD_CLAUSE(25) := ITEM_VALUE(1);

                NEW_CLAUSE(25) := ITEM_VALUE(1);

    when 'G' => OLD_CLAUSE(27) := ITEM_VALUE(1);

                NEW_CLAUSE(27) := ITEM_VALUE(1);

    when 'H' => OLD_CLAUSE(29) := ITEM_VALUE(1);

                NEW_CLAUSE(29) := ITEM_VALUE(1);

    when 'I' => OLD_CLAUSE(31) := ITEM_VALUE(1);

                NEW_CLAUSE(31) := ITEM_VALUE(1);

    when 'J' => OLD_CLAUSE(33) := ITEM_VALUE(1);

                NEW_CLAUSE(33) := ITEM_VALUE(1);

    when 'K' => OLD_CLAUSE(35) := ITEM_VALUE(1);

                NEW_CLAUSE(35) := ITEM_VALUE(1);

    when 'L' => OLD_CLAUSE(37) := ITEM_VALUE(1);

                NEW_CLAUSE(37) := ITEM_VALUE(1);

    when 'M' => OLD_CLAUSE(39) := ITEM_VALUE(1);

                NEW_CLAUSE(39) := ITEM_VALUE(1);

    when others => raise CONSTRAINT_ERROR;
  end case;

end;
```

```
  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
   ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE at " &
    "EXTRACT_ONE_PERIOD");
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at EXTRACT_ONE_PERIOD");
-------------------------------------------------------------------
end EXTRACT_ONE_PERIOD;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Extract two periods gets the selected periods and day from the
-- result of a query
--
-------------------------------------------------------------------
procedure EXTRACT_TWO_PERIODS
-------------------------------------------------------------------
(PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                         DAY_TYPE;
 OLD_P1_CLAUSE   : in out STANDARD.
                         STRING;
 OLD_P2_CLAUSE   : in out STANDARD.
                         STRING;
 NEW_P1_CLAUSE   : in out STANDARD.
                         STRING;
 NEW_P2_CLAUSE   : in out STANDARD.
                         STRING;
 VALUE_LIST      : in out SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING_LIST_PACKAGE.
                         LIST_TYPE) is
-------------------------------------------------------------------

  ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.
                         STRING;
  VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.
                         STRING;

  CONSTRAINT_ERROR : exception;

-------------------------------------------------------------------
begin

  -- List format -> A  m o n  B  p 3 C 0 D x E x .. M o which
  -- formed query

  for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
```

```
                        LENGTH_OF
                        (LIST => VALUE_LIST) / 2
                        -- two at a time
loop

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING_LIST_PACKAGE.
  GET_FROM_FRONT_OF
  (LIST      => VALUE_LIST,
   THIS_ITEM => ITEM_DYNAMIC_STRING);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING_LIST_PACKAGE.
  GET_FROM_FRONT_OF
  (LIST      => VALUE_LIST,
   THIS_ITEM => VALUE_DYNAMIC_STRING);

  declare

    ITEM  : PERIOD_DETAILS_TYPE :=
            PERIOD_DETAILS_TYPE'VALUE
            (SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             SUBSTRING_OF
             (THE_STRING => ITEM_DYNAMIC_STRING));

    ITEM_VALUE : STANDARD.
                 STRING
                 (1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING.
                       LENGTH_OF
                       (THE_STRING => VALUE_DYNAMIC_STRING))) :=
                 SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => VALUE_DYNAMIC_STRING);
  begin

    case ITEM is

      when DAY => THIS_DAY := TIMETABLE_TYPES_PACKAGE.
                              DAY_TYPE'VALUE
                              (ITEM_VALUE);

                  OLD_P1_CLAUSE(8 .. 10) := ITEM_VALUE;
                  NEW_P1_CLAUSE(8 .. 10) := ITEM_VALUE;
                  OLD_P2_CLAUSE(8 .. 10) := ITEM_VALUE;
                  NEW_P2_CLAUSE(8 .. 10) := ITEM_VALUE;

      when FIRST_PERIOD => PERIOD_1 := TIMETABLE_TYPES_PACKAGE.
                                       PERIOD_NUMBER_TYPE'VALUE
                                       (ITEM_VALUE(2 .. 2));

                  OLD_P1_CLAUSE(12 .. 13) := ITEM_VALUE;
                  NEW_P1_CLAUSE(12 .. 13) := ITEM_VALUE;

      when SECOND_PERIOD => PERIOD_2 :=
                            TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE'VALUE
                            (ITEM_VALUE(2 .. 2));
```

```
                                OLD_P2_CLAUSE(12 .. 13) :=
                                ITEM_VALUE;
                                NEW_P2_CLAUSE(12 .. 13) :=
                                ITEM_VALUE;

        when P1W1 .. P2W11 =>
          declare

            PERIOD : POSITIVE :=
            POSITIVE'VALUE
            (SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             SUBSTRING_OF
             (THE_STRING => ITEM_DYNAMIC_STRING,
                            FROM_THE_POSITION => 2,
                            TO_THE_POSITION   => 2));

            WEEK : POSITIVE :=
            POSITIVE'VALUE
            (SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             SUBSTRING_OF
             (THE_STRING =>
              ITEM_DYNAMIC_STRING,
              FROM_THE_POSITION => 4,
              TO_THE_POSITION   => SYSTEM_TYPES_PACKAGE.
                                   DYNAMIC_STRING.
                                   LENGTH_OF
                                   (THE_STRING =>
                                    ITEM_DYNAMIC_STRING)));

          begin
            case PERIOD is
              when 1 =>
                OLD_P1_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
                NEW_P1_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
              when 2 =>
                OLD_P2_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
                NEW_P2_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
              when others => raise CONSTRAINT_ERROR;
            end case;
          end;
        when others => raise CONSTRAINT_ERROR;
      end case;

    end;

  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE at " &
     "EXTRACT_TWO_PERIODS");
  when OTHERS =>
```

```
      TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                       "at EXTRACT_TWO_PERIODS");
  ------------------------------------------------------------------
  end EXTRACT_TWO_PERIODS;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  --
  -- Extract three periods gets the selected periods and day from
the
  -- result of a query
  --
  ------------------------------------------------------------------
  procedure EXTRACT_THREE_PERIODS
  ------------------------------------------------------------------
-
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   PERIOD_3        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   OLD_P1_CLAUSE   : in out STANDARD.
                           STRING;
   OLD_P2_CLAUSE   : in out STANDARD.
                           STRING;
   OLD_P3_CLAUSE   : in out STANDARD.
                           STRING;
   NEW_P1_CLAUSE   : in out STANDARD.
                           STRING;
   NEW_P2_CLAUSE   : in out STANDARD.
                           STRING;
   NEW_P3_CLAUSE   : in out STANDARD.
                           STRING;
   VALUE_LIST      : in out SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LIST_TYPE) is
  ------------------------------------------------------------------

    ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;
    VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;

    CONSTRAINT_ERROR : exception;

  ------------------------------------------------------------------
  begin

    -- List format -> A  m o n  B  p 3 C 0 D x E x .. M o which
    -- formed query

    for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LENGTH_OF
```

```
                        (LIST => VALUE_LIST) / 2
                         -- two at a time
loop

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING_LIST_PACKAGE.
  GET_FROM_FRONT_OF
  (LIST      => VALUE_LIST,
   THIS_ITEM => ITEM_DYNAMIC_STRING);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING_LIST_PACKAGE.
  GET_FROM_FRONT_OF
  (LIST      => VALUE_LIST,
   THIS_ITEM => VALUE_DYNAMIC_STRING);

  declare

    ITEM  : PERIOD_DETAILS_TYPE :=
            PERIOD_DETAILS_TYPE'VALUE
            (SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             SUBSTRING_OF
             (THE_STRING => ITEM_DYNAMIC_STRING));

    ITEM_VALUE : STANDARD.
                 STRING
                 (1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      LENGTH_OF
                      (THE_STRING => VALUE_DYNAMIC_STRING))) :=
                 SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => VALUE_DYNAMIC_STRING);
  begin

    case ITEM is

      when DAY => THIS_DAY := TIMETABLE_TYPES_PACKAGE.
                              DAY_TYPE'VALUE
                              (ITEM_VALUE);

                  OLD_P1_CLAUSE(8 .. 10) := ITEM_VALUE;
                  NEW_P1_CLAUSE(8 .. 10) := ITEM_VALUE;
                  OLD_P2_CLAUSE(8 .. 10) := ITEM_VALUE;
                  NEW_P2_CLAUSE(8 .. 10) := ITEM_VALUE;
                  OLD_P3_CLAUSE(8 .. 10) := ITEM_VALUE;
                  NEW_P3_CLAUSE(8 .. 10) := ITEM_VALUE;


      when FIRST_PERIOD => PERIOD_1 := TIMETABLE_TYPES_PACKAGE.
                                       PERIOD_NUMBER_TYPE'VALUE
                                       (ITEM_VALUE(2 .. 2));

                           OLD_P1_CLAUSE(12 .. 13) := ITEM_VALUE;
                           NEW_P1_CLAUSE(12 .. 13) := ITEM_VALUE;

      when SECOND_PERIOD => PERIOD_2 := TIMETABLE_TYPES_PACKAGE.
                                        PERIOD_NUMBER_TYPE'
```

```
                                        VALUE
                                        (ITEM_VALUE(2 .. 2));

                          OLD_P2_CLAUSE(12 .. 13) :=
                          ITEM_VALUE;
                          NEW_P2_CLAUSE(12 .. 13) :=
                          ITEM_VALUE;

when THIRD_PERIOD => PERIOD_3 := TIMETABLE_TYPES_PACKAGE.
                                 PERIOD_NUMBER_TYPE'VALUE
                                 (ITEM_VALUE(2 .. 2));

                          OLD_P3_CLAUSE(12 .. 13) :=
                          ITEM_VALUE;
                          NEW_P3_CLAUSE(12 .. 13) :=
                          ITEM_VALUE;

when P1W1 .. P3W11 =>

  declare

    PERIOD : POSITIVE :=
             POSITIVE'VALUE
             (SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.
              SUBSTRING_OF
              (THE_STRING =>
               ITEM_DYNAMIC_STRING,
               FROM_THE_POSITION => 2,
               TO_THE_POSITION   => 2));

    WEEK : POSITIVE :=
           POSITIVE'VALUE
           (SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            SUBSTRING_OF
            (THE_STRING =>
             ITEM_DYNAMIC_STRING,
             FROM_THE_POSITION => 4,
             TO_THE_POSITION   =>
             SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             LENGTH_OF
             (THE_STRING =>
              ITEM_DYNAMIC_STRING)));

  begin
    case PERIOD is
      when 1 => OLD_P1_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
                NEW_P1_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
      when 2 => OLD_P2_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
                NEW_P2_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
      when 3 => OLD_P3_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
                NEW_P3_CLAUSE(17 + 2 * WEEK) :=
                ITEM_VALUE(1);
```

```
              when others => raise CONSTRAINT_ERROR;
            end case;
          end;
        when others => raise CONSTRAINT_ERROR;
      end case;

    end;

  end loop;
-------------------------------------------------------------------
exception
 when CONSTRAINT_ERROR =>
  TEXT_IO.
  PUT_LINE("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE at " &
           "EXTRACT_THREE_PERIODS");
 when OTHERS =>
  TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                   "at EXTRACT_THREE_PERIODS");
-------------------------------------------------------------------
end EXTRACT_THREE_PERIODS;
-------------------------------------------------------------------


-------------------------------------------------------------------
function DEGREE
-------------------------------------------------------------------
(ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;
  NODE        : in NATURAL) return CHARACTER is
-------------------------------------------------------------------
begin

  return SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         ITEM_OF
         (THE_STRING => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING_LIST_PACKAGE.
                        ITEM_AT
                        (LIST        => ACTIVITY_PTR.
                                        LIST_OF_COMMON_ACTIVITIES,
                         NODE_NUMBER => NODE),
          AT_THE_POSITION => 1);
-------------------------------------------------------------------
exception
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at DEGREE");
-------------------------------------------------------------------
end DEGREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Common modules available checks all common modules to see
-- whether they are free
--
-------------------------------------------------------------------
function COMMON_MODULES_AVAILABLE
-------------------------------------------------------------------
```

```
(PERIOD_QUERY_STRING : in STANDARD.
                       STRING;
 ACTIVITY_PTR        : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE)
return BOOLEAN is
-------------------------------------------------------------------
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  for COMMON_MODULE in 1 .. SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LENGTH_OF
                            (LIST => ACTIVITY_PTR.
                                     LIST_OF_COMMON_ACTIVITIES)
  loop

    declare

      ESE_PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                         LOGIC_KB.
                         KB_RECORD;

      IT_PERIOD_QUERY :  IT_PERIOD_INFERENCE_PACKAGE.
                         LOGIC_KB.
                         KB_RECORD;

      VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING_LIST_PACKAGE.
                   LIST_TYPE;

      RESPONSE : SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;

    begin

      case SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           ITEM_OF
           (THE_STRING      => ACTIVITY_PTR.
                               FACT,
            AT_THE_POSITION => 1) is

       when 'E' => case DEGREE
                       (ACTIVITY_PTR => ACTIVITY_PTR,
                        NODE         => COMMON_MODULE) is
                   when 'E' => null;

                   when 'I' => IT_PERIOD_INFERENCE_PACKAGE.
                               LOGIC_KB.
                               ASSERT
                               (IN_CLAUSE => PERIOD_QUERY_STRING,
                               KB         => IT_PERIOD_QUERY);

                               -- Ask the query

                               IT_PERIOD_INFERENCE_PACKAGE.
                               SOLVE.
                               START
```

```
                                    (THIS_QUERY => IT_PERIOD_QUERY,
                                     THIS_KB    => IT_PERIOD_KB);

                                     -- Get the result

                                     IT_PERIOD_INFERENCE_PACKAGE.
                                     CONTROL.
                                     GET_RESULT
                                     (OUT_LIST => VALUE_LIST);

                                     -- Get response

                                     SYSTEM_TYPES_PACKAGE.
                                     DYNAMIC_STRING_LIST_PACKAGE.
                                     GET_FROM_FRONT_OF
                                     (LIST      => VALUE_LIST,
                                      THIS_ITEM => RESPONSE);

                                       -- Return if No

                                       if SYSTEM_TYPES_PACKAGE.
                                          DYNAMIC_STRING.
                                          IS_EQUAL
                                          (LEFT => RESPONSE,
                                           RIGHT => "No") then

                                          return FALSE;

                                        end if;

                                        -- Release inference engine

                                        IT_PERIOD_INFERENCE_PACKAGE.
                                        CONTROL.
                                        NO_MORE;

                        when others => raise DEGREE_ERROR;

                    end case;

    when 'I' => case DEGREE
                        (ACTIVITY_PTR => ACTIVITY_PTR,
                         NODE         => COMMON_MODULE) is

                    when 'I' => null;

                    when 'E' => ESE_PERIOD_INFERENCE_PACKAGE.
                                LOGIC_KB.
                                ASSERT
                               (IN_CLAUSE => PERIOD_QUERY_STRING,
                                KB        => ESE_PERIOD_QUERY);

                                -- Ask the query

                                ESE_PERIOD_INFERENCE_PACKAGE.
                                SOLVE.
                                START
                                (THIS_QUERY => ESE_PERIOD_QUERY,
                                 THIS_KB    => ESE_PERIOD_KB);
```

```
                                    -- Get the result

                                    ESE_PERIOD_INFERENCE_PACKAGE.
                                    CONTROL.
                                    GET_RESULT
                                    (OUT_LIST => VALUE_LIST);

                                    -- Get response

                                    SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING_LIST_PACKAGE.
                                    GET_FROM_FRONT_OF
                                    (LIST      => VALUE_LIST,
                                     THIS_ITEM => RESPONSE);

                                      -- Return if No

                                      if SYSTEM_TYPES_PACKAGE.
                                         DYNAMIC_STRING.
                                         IS_EQUAL
                                         (LEFT => RESPONSE,
                                          RIGHT => "No") then

                                         return FALSE;

                                       end if;

                                       -- Release inference engine

                                       ESE_PERIOD_INFERENCE_PACKAGE.
                                       CONTROL.
                                       NO_MORE;

                        when others => raise DEGREE_ERROR;

                    end case;
          when others => raise DEGREE_ERROR;
        end case;
      end;
    end loop;

    -- All modules are free for these periods

    return TRUE;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE("Exception raised in PERIOD_KS_PACKAGE at" &
                     "COMMON_MODULES_AVAILABLE");
    return FALSE;
  when OTHERS =>
    TEXT_IO.PUT_LINE("Exception OTHERS in PERIOD_KS_PACKAGE " &
                     "at COMMON_MODULES_AVAILABLE");
    return FALSE;
-------------------------------------------------------------------
end COMMON_MODULES_AVAILABLE;
-------------------------------------------------------------------


-------------------------------------------------------------------
```

```
--
-- Adjust common_module_periods retracts and asserts the periods
-- taken by
-- lead module
--
-------------------------------------------------------------------
procedure ADJUST_COMMON_MODULE_PERIODS
-------------------------------------------------------------------
(DAY             : in STANDARD.
                      STRING;
 PERIOD          : in STANDARD.
                      STRING;
 WEEKS_REQUIRED  : in TIMETABLE_TYPES_PACKAGE.
                      WEEK_ARRAY_TYPE;
 ACTIVITY_PTR    : in TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  PERIODS_NOT_AVAILABLE : exception;
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  for COMMON_MODULE in 1 .. SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LENGTH_OF
                            (LIST => ACTIVITY_PTR.
                                     LIST_OF_COMMON_ACTIVITIES)
  loop

  declare

    ESE_PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                       LOGIC_KB.
                       KB_RECORD;

    IT_PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                       LOGIC_KB.
                       KB_RECORD;

    VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING_LIST_PACKAGE.
                 LIST_TYPE;

    PERIOD_QUERY_STRING : STANDARD.
                          STRING(1 .. 42) :=
                          "period(" & DAY & "," & PERIOD &
                          ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

    OLD_CLAUSE_STRING : STANDARD.
                        STRING(1 .. 42) :=
                        "period(" & DAY & "," & PERIOD &
                        ",wks(o,o,o,o,o,o,o,o,o,o,o)).";

    NEW_CLAUSE_STRING : STANDARD.
                        STRING(1 .. 42) :=
                        "period(" & DAY & "," & PERIOD &
                        ",wks(o,o,o,o,o,o,o,o,o,o,o)).";

  begin
```

```
case SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     ITEM_OF
     (THE_STRING => ACTIVITY_PTR.
                    FACT,
      AT_THE_POSITION => 1) is

  when 'E' => case DEGREE
                  (ACTIVITY_PTR => ACTIVITY_PTR,
                   NODE         => COMMON_MODULE) is

                when 'E' => null;

                when 'I' => IT_PERIOD_INFERENCE_PACKAGE.
                            LOGIC_KB.
                            ASSERT
                            (IN_CLAUSE => PERIOD_QUERY_STRING,
                             KB        => IT_PERIOD_QUERY);

                            -- Ask

                            IT_PERIOD_INFERENCE_PACKAGE.
                            SOLVE.
                            START
                            (THIS_QUERY => IT_PERIOD_QUERY,
                             THIS_KB    => IT_PERIOD_KB);

                            -- Get result

                            IT_PERIOD_INFERENCE_PACKAGE.
                            CONTROL.
                            GET_RESULT
                            (OUT_LIST => VALUE_LIST);

                            -- Check for "no"

                            if SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               IS_EQUAL
                               (LEFT  => "No",
                                RIGHT =>
                                SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING_LIST_PACKAGE.
                                ITEM_AT
                                (LIST        => VALUE_LIST,
                                 NODE_NUMBER => 1)) then

                              raise PERIODS_NOT_AVAILABLE;

                            end if;

                            -- Should be no more

                            IT_PERIOD_INFERENCE_PACKAGE.
                            CONTROL.
                            NO_MORE;

                            -- Analyse response

                            EXTRACT_WEEKS
```

```
                                    (WEEKS_REQUIRED => WEEKS_REQUIRED,
                                     OLD_CLAUSE     =>
                                     OLD_CLAUSE_STRING,
                                     NEW_CLAUSE     =>
                                     NEW_CLAUSE_STRING,
                                     VALUE_LIST     => VALUE_LIST);

                                    IT_PERIOD_INFERENCE_PACKAGE.
                                    LOGIC_KB.
                                    RETRACT
                                    (CLAUSE => OLD_CLAUSE_STRING,
                                     KB     => IT_PERIOD_KB);

                                     if NEW_CLAUSE_STRING(18 .. 40) =
                                        "(x,x,x,x,x,x,x,x,x,x)" then

                                       IT_PERIOD_INFERENCE_PACKAGE.
                                       LOGIC_KB.
                                       ASSERT
                                      (IN_CLAUSE => NEW_CLAUSE_STRING,
                                       KB        => IT_PERIOD_KB);

                                     else

                                       IT_PERIOD_INFERENCE_PACKAGE.
                                       LOGIC_KB.
                                       ASSERT
                                       (IN_CLAUSE  =>
                                        NEW_CLAUSE_STRING,
                                        KB         => IT_PERIOD_KB,
                                        AT_BACK_OF => FALSE);

                                     end if;

                  when others => raise DEGREE_ERROR;

               end case;

   when 'I' => case DEGREE
                          (ACTIVITY_PTR => ACTIVITY_PTR,
                           NODE         => COMMON_MODULE) is

                  when 'E' => ESE_PERIOD_INFERENCE_PACKAGE.
                              LOGIC_KB.
                              ASSERT
                              (IN_CLAUSE => PERIOD_QUERY_STRING,
                               KB        => ESE_PERIOD_QUERY);

                              -- Ask

                              ESE_PERIOD_INFERENCE_PACKAGE.
                              SOLVE.
                              START
                              (THIS_QUERY => ESE_PERIOD_QUERY,
                               THIS_KB    => ESE_PERIOD_KB);

                              -- Get result

                              ESE_PERIOD_INFERENCE_PACKAGE.
                              CONTROL.
```

```
GET_RESULT
(OUT_LIST => VALUE_LIST);

-- Check for "no"

if SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   IS_EQUAL
   (LEFT  => "No",
    RIGHT =>
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    ITEM_AT
    (LIST        => VALUE_LIST,
     NODE_NUMBER => 1)) then

  raise PERIODS_NOT_AVAILABLE;

end if;

-- Should be no more

ESE_PERIOD_INFERENCE_PACKAGE.
CONTROL.
NO_MORE;

-- Analyse response

EXTRACT_WEEKS
(WEEKS_REQUIRED => WEEKS_REQUIRED,
 OLD_CLAUSE     =>
 OLD_CLAUSE_STRING,
 NEW_CLAUSE     =>
 NEW_CLAUSE_STRING,
 VALUE_LIST     => VALUE_LIST);

ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_CLAUSE_STRING,
 KB     => ESE_PERIOD_KB);

 if NEW_CLAUSE_STRING(18 .. 40) =
    "(x,x,x,x,x,x,x,x,x,x)" then

   ESE_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
  (IN_CLAUSE => NEW_CLAUSE_STRING,
   KB        => ESE_PERIOD_KB);

 else

   ESE_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (IN_CLAUSE  =>
    NEW_CLAUSE_STRING,
    KB         => ESE_PERIOD_KB,
    AT_BACK_OF => FALSE);
```

```
                                    end if;
                        when 'I' => null;
                        when others => raise DEGREE_ERROR;
                    end case;
        when others => raise DEGREE_ERROR;
      end case;
    end;
    end loop;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR in PERIOD_KS_PACKAGE at" &
     "ADJUST_COMMON_MODULE_PERIODS");
  when PERIODS_NOT_AVAILABLE =>
    TEXT_IO.PUT_LINE
    ("Exception PERIODS_NOT_AVAILABLE in PERIOD_KS_PACKAGE at" &
     "ADJUST_COMMON_MODULE_PERIODS");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_KS_PACKAGE " &
     "at ADJUST_COMMON_MODULE_PERIODS");
-------------------------------------------------------------------
end ADJUST_COMMON_MODULE_PERIODS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Add_period_details adds the select periods to the blackboard
--
-------------------------------------------------------------------
procedure ADD_PERIOD_DETAILS
-------------------------------------------------------------------
(THIS_DAY        : in      TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
 FIRST_PERIOD    : in      TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
 LAST_PERIOD     : in      TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
 WEEKS_REQUIRED  : in      TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
 ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD      : in out  TIMETABLE_BLACKBOARD_PACKAGE.
                           TIMETABLE_BLACKBOARD.
                           BLACKBOARD_TYPE) is
-------------------------------------------------------------------

  ROOT : SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         STRING;
```

```
   RESOURCE : SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.
              STRING;

   ACTIVITY : SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.
              STRING;

NUMBER_OF_STAFF : STANDARD.NATURAL := SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING_LIST_PACKAGE.
                                      LENGTH_OF
                                      (LIST => ACTIVITY_PTR.
                                       STAFF_LIST);

 ROOMS            : SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING_ARRAY(1 .. NUMBER_OF_STAFF);

 PERIODS : SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING_ARRAY(FIRST_PERIOD ..  LAST_PERIOD);

 NUMBER_OF_PERIODS : POSITIVE := LAST_PERIOD - FIRST_PERIOD + 1;

 GROUP_SIZE : POSITIVE := 1;

 SELECTED : BOOLEAN := FALSE;

-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "ROOT",
   TO_THE_STRING      => ROOT);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "RESOURCE",
   TO_THE_STRING      => RESOURCE);

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "ACTIVITY",
   TO_THE_STRING      => ACTIVITY);

  -- Fill period array

  for PERIOD in FIRST_PERIOD .. LAST_PERIOD
  loop
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_SUBSTRING => "p",
     TO_THE_STRING      => PERIODS(PERIOD));

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
```

```
   APPEND
   (TO_THE_STRING => PERIODS(PERIOD),
    THE_SUBSTRING => TIMETABLE_TYPES_PACKAGE.
                     PERIOD_NUMBER_TYPE'
                     IMAGE
                     (PERIOD)(2 .. TIMETABLE_TYPES_PACKAGE.
                                   PERIOD_NUMBER_TYPE'
                                   IMAGE
                                   (PERIOD)'LAST));
end loop;

-- Record first period in activity

ACTIVITY_PTR.FIRST_PERIOD := FIRST_PERIOD;
ACTIVITY_PTR.FIRST_PERIOD_STRING := "p" &
                                    TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE(FIRST_PERIOD)(2);

if NUMBER_OF_STAFF /= 0 and then NUMBER_OF_STAFF /= 1 then

  GROUP_SIZE := ACTIVITY_PTR.STUDENTS / NUMBER_OF_STAFF;
else

  GROUP_SIZE := ACTIVITY_PTR.STUDENTS;
end if;

ROOM_PACKAGE.
ALLOCATE_ROOMS
(ROOMS          => ROOMS,
 PERIODS        => PERIODS,
 THIS_DAY       => THIS_DAY,
 WEEKS_REQUIRED => WEEKS_REQUIRED,
 ACTIVITY_PTR   => ACTIVITY_PTR,
 SELECTED       => SELECTED);

for PERIOD_NUMBER in FIRST_PERIOD .. LAST_PERIOD
loop

  -- Add activity frame

  TIMETABLE_TYPES_PACKAGE.
  PERIOD_FRAME_BASE_PACKAGE.
  ADD_FRAME
  (FRAME_NAME         => ACTIVITY_PTR.
                         FACT,
   SUPERCLASS_NAME    => ROOT,
   FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_ITEM
                         (BLACKBOARD  => BLACKBOARD,
                          LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                         DAYS,
                          ITEM_INDEX  => CONVERT(THIS_DAY)).
                          PERIOD(PERIOD_NUMBER).
                          PERIOD_DETAIL_FRAMES);

  -- Add common activity frames

  for ACTIVITY_NUMBER in  1 .. SYSTEM_TYPES_PACKAGE.
```

```
                                DYNAMIC_STRING_LIST_PACKAGE.
                                LENGTH_OF
                                (LIST =>
                                 ACTIVITY_PTR.
                                 LIST_OF_COMMON_ACTIVITIES)
loop

  TIMETABLE_TYPES_PACKAGE.
  PERIOD_FRAME_BASE_PACKAGE.
  ADD_FRAME
  (FRAME_NAME         => SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING_LIST_PACKAGE.
                         ITEM_AT
                         (LIST         =>
                          ACTIVITY_PTR.
                          LIST_OF_COMMON_ACTIVITIES,
                          NODE_NUMBER => ACTIVITY_NUMBER),
   SUPERCLASS_NAME    => ACTIVITY_PTR.
                         FACT,
   FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_ITEM
                         (BLACKBOARD   => BLACKBOARD,
                          LEVEL_INDEX =>
                          TIMETABLE_TYPES_PACKAGE.
                          DAYS,
                          ITEM_INDEX   => CONVERT(THIS_DAY)).
                        PERIOD(PERIOD_NUMBER).
                        PERIOD_DETAIL_FRAMES);

end loop;

-- Add resource slot

TIMETABLE_TYPES_PACKAGE.
PERIOD_FRAME_BASE_PACKAGE.
ADD_SLOT
(SLOT_NAME          => RESOURCE,
 FRAME_NAME         => ACTIVITY_PTR.
                       FACT,
 FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_ITEM
                       (BLACKBOARD   => BLACKBOARD,
                        LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                       DAYS,
                        ITEM_INDEX   => CONVERT(THIS_DAY)).
                      PERIOD(PERIOD_NUMBER).
                      PERIOD_DETAIL_FRAMES);

-- Add activity slot

TIMETABLE_TYPES_PACKAGE.
PERIOD_FRAME_BASE_PACKAGE.
ADD_SLOT
(SLOT_NAME          => ACTIVITY,
 FRAME_NAME         => ACTIVITY_PTR.
                       FACT,
 FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
```

```
                        BLACKBOARD_ITEM
                        (BLACKBOARD  => BLACKBOARD,
                         LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                        DAYS,
                         ITEM_INDEX  => CONVERT(THIS_DAY)).
                        PERIOD(PERIOD_NUMBER).
                        PERIOD_DETAIL_FRAMES);

-- Add staff facets

  declare

    RESOURCE_FACET_PTR : TIMETABLE_TYPES_PACKAGE.
                         PERIOD_FACET_RECORD_PTR_TYPE;
  begin

    for STAFF_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_LIST_PACKAGE.
                             LENGTH_OF
                             (LIST => ACTIVITY_PTR.
                                      STAFF_LIST)
    loop

      RESOURCE_FACET_PTR := new TIMETABLE_TYPES_PACKAGE.
                                PERIOD_FACET_RECORD_TYPE
                                (TIMETABLE_TYPES_PACKAGE.
                                 RESOURCE);

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_STRING => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          ITEM_AT
                          (LIST => ACTIVITY_PTR.
                                   STAFF_LIST,
                           NODE_NUMBER => STAFF_NUMBER),
       TO_THE_STRING   => RESOURCE_FACET_PTR.
                          INITIALS);

      -- Add week code

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_STRING => ACTIVITY_PTR.
                          FREQUENCY,
       TO_THE_STRING   => RESOURCE_FACET_PTR.
                          WEEK_CODE);

      -- Add converted week code

      RESOURCE_FACET_PTR.
      WEEK_ARRAY := WEEKS_REQUIRED;

      -- Add the room

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
```

```
        (FROM_THE_STRING => ROOMS(STAFF_NUMBER),
         TO_THE_STRING   => RESOURCE_FACET_PTR.
                            ROOM);

        -- Add facet to blackboard

        TIMETABLE_TYPES_PACKAGE.
        PERIOD_FRAME_BASE_PACKAGE.
        ADD_FACET
        (FACET_PTR          => RESOURCE_FACET_PTR,
         SLOT_NAME          => RESOURCE,
         FRAME_NAME         => ACTIVITY_PTR.
                               FACT,
         FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                               TIMETABLE_BLACKBOARD.
                               BLACKBOARD_ITEM
                               (BLACKBOARD  => BLACKBOARD,
                                LEVEL_INDEX =>
                                TIMETABLE_TYPES_PACKAGE.
                                DAYS,
                                ITEM_INDEX  =>
                                CONVERT(THIS_DAY)).
                                PERIOD(PERIOD_NUMBER).
                                PERIOD_DETAIL_FRAMES);

    end loop;

  end;

-- Add activity facet

declare
  ACTIVITY_FACET_PTR : TIMETABLE_TYPES_PACKAGE.
                       PERIOD_FACET_RECORD_PTR_TYPE;
begin
  ACTIVITY_FACET_PTR := new TIMETABLE_TYPES_PACKAGE.
                            PERIOD_FACET_RECORD_TYPE
                            (TIMETABLE_TYPES_PACKAGE.
                             ACTIVITY);
  ACTIVITY_FACET_PTR.NUMBER       := PERIOD_NUMBER;
  ACTIVITY_FACET_PTR.TOTAL        := LAST_PERIOD;
  ACTIVITY_FACET_PTR.ACTIVITY_PTR := ACTIVITY_PTR;

  TIMETABLE_TYPES_PACKAGE.
  PERIOD_FRAME_BASE_PACKAGE.
  ADD_FACET
  (FACET_PTR          => ACTIVITY_FACET_PTR,
   SLOT_NAME          => ACTIVITY,
   FRAME_NAME         => ACTIVITY_PTR.
                         FACT,
   FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_ITEM
                         (BLACKBOARD  => BLACKBOARD,
                          LEVEL_INDEX =>
                          TIMETABLE_TYPES_PACKAGE.
                          DAYS,
                          ITEM_INDEX  => CONVERT(THIS_DAY)).
                          PERIOD(PERIOD_NUMBER).
                          PERIOD_DETAIL_FRAMES);
```

```
    end;

    -- Connect activity to the period

    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DAYS,
                    ITEM_INDEX  => CONVERT(THIS_DAY)).
                    PERIOD(PERIOD_NUMBER).
                    SUPPORTERS,
     THIS_ITEM => ACTIVITY_PTR);

  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_3_PACKAGE at " &
     "ADD_PERIOD_DETAILS");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_3_PACKAGE at " &
     "ADD_PERIOD_DETAILS");
-------------------------------------------------------------------
end ADD_PERIOD_DETAILS;
-------------------------------------------------------------------


-------------------------------------------------------------------
function CONVERT
-------------------------------------------------------------------
  (DAY : in TIMETABLE_TYPES_PACKAGE.
            DAY_TYPE) return TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_ITEM_TYPE is
-------------------------------------------------------------------
  DAY_ERROR : exception;
-------------------------------------------------------------------
begin

  case DAY is

    when TIMETABLE_TYPES_PACKAGE.mon =>
         return TIMETABLE_TYPES_PACKAGE.MONDAY;
    when TIMETABLE_TYPES_PACKAGE.tue =>
         return TIMETABLE_TYPES_PACKAGE.TUESDAY;
    when TIMETABLE_TYPES_PACKAGE.wed =>
         return TIMETABLE_TYPES_PACKAGE.WEDNESDAY;
    when TIMETABLE_TYPES_PACKAGE.thu =>
         return TIMETABLE_TYPES_PACKAGE.THURSDAY;
    when TIMETABLE_TYPES_PACKAGE.fri =>
         return TIMETABLE_TYPES_PACKAGE.FRIDAY;
    when others                      =>
         raise DAY_ERROR;
```

```
    end case;

  exception
    when DAY_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception DAY_ERROR in PERIOD_KS_PACKAGE at " &
       "CONVERT_1");
    when others =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_KS_PACKAGE at " &
       "CONVERT_1");
  ------------------------------------------------------------------
  end CONVERT;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  function CONVERT
  ------------------------------------------------------------------
  (DAY : in TIMETABLE_TYPES_PACKAGE.
            TIMETABLE_ITEM_TYPE) return TIMETABLE_TYPES_PACKAGE.
                                       DAY_TYPE is
  ------------------------------------------------------------------
    DAY_ERROR : exception;
  ------------------------------------------------------------------
  begin

    case DAY is
      when TIMETABLE_TYPES_PACKAGE.MONDAY    =>
           return TIMETABLE_TYPES_PACKAGE.MON;
      when TIMETABLE_TYPES_PACKAGE.TUESDAY   =>
           return TIMETABLE_TYPES_PACKAGE.TUE;
      when TIMETABLE_TYPES_PACKAGE.WEDNESDAY =>
           return TIMETABLE_TYPES_PACKAGE.WED;
      when TIMETABLE_TYPES_PACKAGE.THURSDAY  =>
           return TIMETABLE_TYPES_PACKAGE.THU;
      when TIMETABLE_TYPES_PACKAGE.FRIDAY    =>
           return TIMETABLE_TYPES_PACKAGE.FRI;
      when others                            =>
           raise DAY_ERROR;
    end case;
  ------------------------------------------------------------------
  exception
    when DAY_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception DAY_ERROR in PERIOD_KS_PACKAGE at " &
       "CONVERT_2");
    when others =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_KS_PACKAGE at " &
       "CONVERT_2");
  ------------------------------------------------------------------
    end CONVERT;
  ------------------------------------------------------------------

------------------------------------------------------------------
end PERIOD_3_PACKAGE;
------------------------------------------------------------------
```

Period 4

```
------------------------------------------------------------------------
--
-- Unit      : PERIOD_4_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function : This package provides period utilities
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE;
use  PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE;
------------------------------------------------------------------------
package PERIOD_4_PACKAGE is
------------------------------------------------------------------------
  ----------------------------------------------------------------------
  procedure EXTRACT_WEEKS
  ----------------------------------------------------------------------
  (VALUE_LIST : in out SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING_LIST_PACKAGE.
                       LIST_TYPE;
   OLD_CLAUSE : in out STANDARD.STRING);
  ----------------------------------------------------------------------
  function OCCUPIED_PERIOD_ALREADY_PREFERED
  (DAY        : in STANDARD.STRING;
   PERIOD     : in STANDARD.STRING;
   BLACKBOARD : in TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_TYPE) return BOOLEAN;

  procedure RESCHEDULE_OCCUPIED_PERIOD_ACTIVITIES
  (DAY        : in STANDARD.STRING;
   PERIOD     : in STANDARD.STRING;
   BLACKBOARD : in TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_TYPE);

  procedure RESET_KNOWLEDGE_BASES
  (DAY        : in STANDARD.STRING;
   PERIOD     : in STANDARD.STRING;
   BLACKBOARD : in TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_TYPE);
------------------------------------------------------------------------
end PERIOD_4_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_4_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield , RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function : This package provides the period utilities
--
-------------------------------------------------------------------
package body PERIOD_4_PACKAGE is
-------------------------------------------------------------------
use TIMETABLE_TYPES_PACKAGE;
  -----------------------------------------------------------------
  procedure EXTRACT_WEEKS
  -----------------------------------------------------------------
  (VALUE_LIST : in out SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING_LIST_PACKAGE.
                       LIST_TYPE;
   OLD_CLAUSE : in out STANDARD.STRING) is
  -----------------------------------------------------------------
  begin
    for ITEM_NUMBER in 1.. SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LENGTH_OF
                           (LIST => VALUE_LIST) / 2
    loop
      declare
        ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
        VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING.
                               STRING;
      begin
        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING_LIST_PACKAGE.
        GET_FROM_FRONT_OF
        (LIST        => VALUE_LIST,
         THIS_ITEM   => ITEM_DYNAMIC_STRING);

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING_LIST_PACKAGE.
        GET_FROM_FRONT_OF
        (LIST        => VALUE_LIST,
         THIS_ITEM   => VALUE_DYNAMIC_STRING);

        declare
          ITEM : STANDARD.CHARACTER :=
                 SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => ITEM_DYNAMIC_STRING)(1);

          ITEM_VALUE : STANDARD.STRING
                       (1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                                      DYNAMIC_STRING.
                                      LENGTH_OF
                                      (THE_STRING =>
                                       VALUE_DYNAMIC_STRING))) :=
                       SYSTEM_TYPES_PACKAGE.
```

```
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => VALUE_DYNAMIC_STRING);
      begin
        case ITEM is
          when 'A' => OLD_CLAUSE(19) := ITEM_VALUE(1);
          when 'B' => OLD_CLAUSE(21) := ITEM_VALUE(1);
          when 'C' => OLD_CLAUSE(23) := ITEM_VALUE(1);
          when 'D' => OLD_CLAUSE(25) := ITEM_VALUE(1);
          when 'E' => OLD_CLAUSE(27) := ITEM_VALUE(1);
          when 'F' => OLD_CLAUSE(29) := ITEM_VALUE(1);
          when 'G' => OLD_CLAUSE(31) := ITEM_VALUE(1);
          when 'H' => OLD_CLAUSE(33) := ITEM_VALUE(1);
          when 'I' => OLD_CLAUSE(35) := ITEM_VALUE(1);
          when 'J' => OLD_CLAUSE(37) := ITEM_VALUE(1);
          when 'K' => OLD_CLAUSE(39) := ITEM_VALUE(1);
          when others => raise CONSTRAINT_ERROR;
        end case;
      end;
    end;
  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_UTIL_PACKAGE at " &
     "EXTRACT_WEEKS");

  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_UTIL_PACKAGE " &
     "at EXTRACT_WEEKS");
-------------------------------------------------------------------
end EXTRACT_WEEKS;
-------------------------------------------------------------------


-------------------------------------------------------------------
function OCCUPIED_PERIOD_ALREADY_PREFERED
-------------------------------------------------------------------
(DAY        : in STANDARD.STRING;
 PERIOD     : in STANDARD.STRING;
 BLACKBOARD : in TIMETABLE_BLACKBOARD_PACKAGE.
               TIMETABLE_BLACKBOARD.
               BLACKBOARD_TYPE) return BOOLEAN is
-------------------------------------------------------------------
  PERIOD_NUMBER : TIMETABLE_TYPES_PACKAGE.
                  PERIOD_NUMBER_TYPE :=
                  TIMETABLE_TYPES_PACKAGE.
                  PERIOD_NUMBER_TYPE'
                  VALUE
                  (PERIOD);

  ITEM_INDEX : TIMETABLE_TYPES_PACKAGE.
               DAY_TYPE :=
               TIMETABLE_TYPES_PACKAGE.
               DAY_TYPE'
               VALUE
               (DAY);
```

```
   ACTIVITY_LIST : TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_LIST_PACKAGE.
                   LIST_TYPE :=
                   TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_ITEM
                   (BLACKBOARD   => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DAYS,
                    ITEM_INDEX   => CONVERT(ITEM_INDEX)).
                   PERIOD(PERIOD_NUMBER).
                   SUPPORTERS;
-------------------------------------------------------------------
begin

   for ACTIVITY_NUMBER in 1 .. TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_LIST_PACKAGE.
                               LENGTH_OF
                               (ACTIVITY_LIST)
   loop
     if TIMETABLE_TYPES_PACKAGE.
        TIMETABLE_LIST_PACKAGE.
        ITEM_AT
        (LIST => ACTIVITY_LIST,
         NODE_NUMBER => ACTIVITY_NUMBER).
        HAS_PREFERENCE then

       return TRUE;

     end if;

   end loop;

   return FALSE;
-------------------------------------------------------------------
exception
   when CONSTRAINT_ERROR =>
     TEXT_IO.PUT_LINE
     ("Exception CONSTRAINT_ERROR in PERIOD_UTIL_PACKAGE at " &
      "OCCUPIED_PERIOD_ALREADY_PREFERED");
   when OTHERS =>
     TEXT_IO.PUT_LINE
     ("Exception OTHERS in PERIOD_UTIL_PACKAGE " &
      "at OCCUPIED_PERIOD_ALREADY_PREFERED");
-------------------------------------------------------------------
end OCCUPIED_PERIOD_ALREADY_PREFERED;
-------------------------------------------------------------------

-------------------------------------------------------------------
procedure RESCHEDULE
-------------------------------------------------------------------
(DAY           : in     STANDARD.STRING;
 PERIOD        : in     STANDARD.STRING;
 BLACKBOARD    : in     TIMETABLE_BLACKBOARD_PACKAGE.
                        TIMETABLE_BLACKBOARD.
                        BLACKBOARD_TYPE;
 SCHEDULE_LIST : in out TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_LIST_PACKAGE.
                        LIST_TYPE) is
-------------------------------------------------------------------
```

```
    PERIOD_NUMBER : TIMETABLE_TYPES_PACKAGE.
                    PERIOD_NUMBER_TYPE :=
                    TIMETABLE_TYPES_PACKAGE.
                    PERIOD_NUMBER_TYPE'
                    VALUE
                    (PERIOD);

    ITEM_INDEX : TIMETABLE_TYPES_PACKAGE.
                 DAY_TYPE :=
                 TIMETABLE_TYPES_PACKAGE.
                 DAY_TYPE'
                 VALUE
                 (DAY);

    ACTIVITY_LIST : TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_LIST_PACKAGE.
                    LIST_TYPE :=
                    TIMETABLE_BLACKBOARD_PACKAGE.
                    TIMETABLE_BLACKBOARD.
                    BLACKBOARD_ITEM
                   (BLACKBOARD  => BLACKBOARD,
                    LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                   DAYS,
                    ITEM_INDEX  => CONVERT(ITEM_INDEX)).
                   PERIOD(PERIOD_NUMBER).
                   SUPPORTERS;

    ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
  ------------------------------------------------------------------
    PERIOD_NUMBER_ERROR : exception;
  ------------------------------------------------------------------
  begin

    -- Collect each activity to reschedule

    for ACTIVITY_NUMBER  in 1 .. TIMETABLE_TYPES_PACKAGE.
                                 TIMETABLE_LIST_PACKAGE.
                                 LENGTH_OF
                                 (ACTIVITY_LIST)
    loop

      ACTIVITY_PTR := TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_LIST_PACKAGE.
                      ITEM_AT
                      (LIST        => ACTIVITY_LIST,
                       NODE_NUMBER => ACTIVITY_NUMBER);

      if not TIMETABLE_TYPES_PACKAGE.
             TIMETABLE_LIST_PACKAGE.
             IS_IN(LIST => SCHEDULE_LIST,
                   ITEM => ACTIVITY_PTR) then

        TIMETABLE_TYPES_PACKAGE.
        TIMETABLE_LIST_PACKAGE.
        PUT_ON_BACK_OF
       (LIST      => SCHEDULE_LIST,
        THIS_ITEM => ACTIVITY_PTR);

        case ACTIVITY_PTR.NUMBER_OF_PERIODS is
```

```
when 1 =>       null;
when 2 =>       if ACTIVITY_PTR.FIRST_PERIOD =
                   PERIOD_NUMBER then

                  RESCHEDULE
                  (DAY           => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER + 1)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

                else

                 RESCHEDULE
                 (DAY            => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER - 1)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

                end if;

when 3 =>       if ACTIVITY_PTR.FIRST_PERIOD =
                   PERIOD_NUMBER then

                 RESCHEDULE
                 (DAY            => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER + 1)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

                  RESCHEDULE
                  (DAY            => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER + 2)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

                elsif ACTIVITY_PTR.FIRST_PERIOD =
                      PERIOD_NUMBER - 1 then

                  RESCHEDULE
                  (DAY           => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER - 1)
```

```
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

                  RESCHEDULE
                  (DAY           => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER + 1)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

               else

                  RESCHEDULE
                  (DAY           => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER - 1)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

                  RESCHEDULE
                  (DAY.          => DAY,
                  PERIOD         => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_NUMBER_TYPE'
                                    IMAGE
                                    (PERIOD_NUMBER - 2)
                                    (2 .. 2),
                  BLACKBOARD     => BLACKBOARD,
                  SCHEDULE_LIST => SCHEDULE_LIST);

               end if;

      when others => raise PERIOD_NUMBER_ERROR;
    end case;

  end if;

end loop;

-- Clear the supporters

TIMETABLE_TYPES_PACKAGE.
TIMETABLE_LIST_PACKAGE.
CLEAR
(LIST => TIMETABLE_BLACKBOARD_PACKAGE.
         TIMETABLE_BLACKBOARD.
         BLACKBOARD_ITEM
         (BLACKBOARD => BLACKBOARD,
          LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                         DAYS,
          ITEM_INDEX  => CONVERT(ITEM_INDEX)).
          PERIOD(PERIOD_NUMBER).
          SUPPORTERS);
```

```
   --Clear the blackboard period

   TIMETABLE_TYPES_PACKAGE.
   PERIOD_FRAME_BASE_PACKAGE.
   CLEAR
   (TIMETABLE_BLACKBOARD_PACKAGE.
    TIMETABLE_BLACKBOARD.
    BLACKBOARD_ITEM
    (BLACKBOARD  => BLACKBOARD,
     LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                    DAYS,
     ITEM_INDEX  => CONVERT(ITEM_INDEX)).
    PERIOD(PERIOD_NUMBER).
    PERIOD_DETAIL_FRAMES);
-------------------------------------------------------------------
exception
  when PERIOD_NUMBER_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception PERIOD_NUMBER_ERROR in PERIOD_4_PACKAGE at " &
     "RESCHEDULE");
-------------------------------------------------------------------
end RESCHEDULE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure RESCHEDULE_OCCUPIED_PERIOD_ACTIVITIES
-------------------------------------------------------------------
(DAY        : in STANDARD.STRING;
 PERIOD     : in STANDARD.STRING;
 BLACKBOARD : in TIMETABLE_BLACKBOARD_PACKAGE.
                 TIMETABLE_BLACKBOARD.
                 BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  SCHEDULE_LIST : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_LIST_PACKAGE.
                  LIST_TYPE;

  ACTIVITY_PTR_PTR : TIMETABLE_TYPES_PACKAGE.
                     TIMETABLE_NODE_PTR_TYPE;
-------------------------------------------------------------------
begin

  -- Collect all activities

  RESCHEDULE
  (DAY           => DAY,
   PERIOD        => PERIOD,
   BLACKBOARD    => BLACKBOARD,
   SCHEDULE_LIST => SCHEDULE_LIST);

  -- Return them all to the event list

  for ACTIVITY_NUMBER  in 1 .. TIMETABLE_TYPES_PACKAGE.
                               TIMETABLE_LIST_PACKAGE.
                               LENGTH_OF
                               (SCHEDULE_LIST)
  loop

    TIMETABLE_TYPES_PACKAGE.
```

```
      TIMETABLE_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST       => TIMETABLE_BLACKBOARD_PACKAGE.
                     TIMETABLE_BLACKBOARD.
                     BLACKBOARD_ITEM
                     (BLACKBOARD  => BLACKBOARD,
                      LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                     EVENT_LISTS,
                      ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                     PERIOD_EVENTS).
                     LIST,
       THIS_ITEM => TIMETABLE_TYPES_PACKAGE.
                     TIMETABLE_LIST_PACKAGE.
                     ITEM_AT
                     (LIST        => SCHEDULE_LIST,
                      NODE_NUMBER => ACTIVITY_NUMBER));

      -- Set all staff free

      declare
        ACTIVITY   : TIMETABLE_TYPES_PACKAGE.
                     TIMETABLE_NODE_PTR_TYPE;
        WEEK_ARRAY : TIMETABLE_TYPES_PACKAGE.
                     WEEK_ARRAY_TYPE;
      begin
        ACTIVITY := TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_LIST_PACKAGE.
                    ITEM_AT
                    (LIST        => SCHEDULE_LIST,
                     NODE_NUMBER => ACTIVITY_NUMBER);
        CONVERT
        (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING.
                       SUBSTRING_OF
                       (THE_STRING => ACTIVITY.FREQUENCY),
         WEEK_ARRAY => WEEK_ARRAY);

        STAFF_PACKAGE.
        MAKE_ALL_STAFF_FREE
        (STAFF_LIST                => ACTIVITY.STAFF_LIST,
         MODULE                    => ACTIVITY.FACT,
         DAY                       => TIMETABLE_TYPES_PACKAGE.
                                      DAY_TYPE'VALUE(DAY),
         PERIOD                    => TIMETABLE_TYPES_PACKAGE.
                                      PERIOD_NUMBER_TYPE'
                                      VALUE(PERIOD),
         WEEK_ARRAY                => WEEK_ARRAY,
         STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                      STAFF_FRAME_BASE_RECORD);
      end;

    end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_4_PACKAGE at " &
     "RESCHEDULE_OCCUPIED_PERIOD_ACTIVITIES");
  when OTHERS =>
    TEXT_IO.PUT_LINE
```

```
    ("Exception OTHERS in PERIOD_4_PACKAGE " &
     "at RESCHEDULE_OCCUPIED_PERIOD_ACTIVITIES");
-------------------------------------------------------------------
end RESCHEDULE_OCCUPIED_PERIOD_ACTIVITIES;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure RESET
-------------------------------------------------------------------
(DAY          : in STANDARD.STRING;
 PERIOD       : in STANDARD.STRING;
 ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD   : in TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  NEW_CLAUSE : STANDARD.STRING(1 .. 42) :=
               "period(" &
               SYSTEM_TYPES_PACKAGE.
               CONVERT(DAY) &
               ",p" &
               PERIOD &
               ",wks(o,o,o,o,o,o,o,o,o,o,o)).";

  OLD_CLAUSE : STANDARD.STRING(1 .. 42) :=
               "period(" &
               SYSTEM_TYPES_PACKAGE.
               CONVERT(DAY) &
               ",p" &
               PERIOD &
               ",wks(o,o,o,o,o,o,o,o,o,o,o)).";

  OLD_CLAUSE_QUERY : STANDARD.STRING(1 .. 42) :=
                     "period(" &
                     SYSTEM_TYPES_PACKAGE.
                     CONVERT(DAY) &
                     ",p" &
                     PERIOD &
                     ",wks(A,B,C,D,E,F,G,H,I,J,K)).";
  ESE_PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                     LOGIC_KB.
                     KB_RECORD;

  IT_PERIOD_QUERY : IT_PERIOD_INFERENCE_PACKAGE.
                    LOGIC_KB.
                    KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;
-------------------------------------------------------------------
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

    case   SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           SUBSTRING_OF
```

```
      (THE_STRING => ACTIVITY_PTR.FACT)(1) is
when 'E' => ESE_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            ASSERT
            (IN_CLAUSE => OLD_CLAUSE_QUERY,
             KB        => ESE_PERIOD_QUERY);

             -- ask

             ESE_PERIOD_INFERENCE_PACKAGE.
             SOLVE.
             START
             (THIS_QUERY => ESE_PERIOD_QUERY,
              THIS_KB    => ESE_PERIOD_KB);

             -- Get result

             ESE_PERIOD_INFERENCE_PACKAGE.
             CONTROL.
             GET_RESULT
             (OUT_LIST => VALUE_LIST);

             -- Release KB

             ESE_PERIOD_INFERENCE_PACKAGE.
             CONTROL.
             NO_MORE;

             -- Get the current weeks fom the response

            EXTRACT_WEEKS
            (VALUE_LIST => VALUE_LIST,
             OLD_CLAUSE => OLD_CLAUSE);

            -- Retract old clause

            ESE_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            RETRACT
            (CLAUSE => OLD_CLAUSE,
             KB     => ESE_PERIOD_KB);

            -- Assert cleared clause

            ESE_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            ASSERT
            (IN_CLAUSE  => NEW_CLAUSE,
             KB         => ESE_PERIOD_KB,
             AT_BACK_OF => FALSE);

when 'I' => IT_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            ASSERT
            (IN_CLAUSE => OLD_CLAUSE_QUERY,
             KB        => IT_PERIOD_QUERY);

             -- ask

             IT_PERIOD_INFERENCE_PACKAGE.
```

```
                    SOLVE.
                    START
                    (THIS_QUERY => IT_PERIOD_QUERY,
                     THIS_KB    => IT_PERIOD_KB);

                    -- Get result

                    IT_PERIOD_INFERENCE_PACKAGE.
                    CONTROL.
                    GET_RESULT
                    (OUT_LIST => VALUE_LIST);

                    -- Release KB

                    IT_PERIOD_INFERENCE_PACKAGE.
                    CONTROL.
                    NO_MORE;

                    -- Get the current weeks fom the response

                   EXTRACT_WEEKS
                   (VALUE_LIST => VALUE_LIST,
                    OLD_CLAUSE => OLD_CLAUSE);

                   -- Retract old clause

                   IT_PERIOD_INFERENCE_PACKAGE.
                   LOGIC_KB.
                   RETRACT
                   (CLAUSE => OLD_CLAUSE,
                    KB     => IT_PERIOD_KB);

                   -- Assert cleared clause

                   IT_PERIOD_INFERENCE_PACKAGE.
                   LOGIC_KB.
                   ASSERT
                   (IN_CLAUSE  => NEW_CLAUSE,
                    KB         => IT_PERIOD_KB,
                    AT_BACK_OF => FALSE);

  when others => raise DEGREE_ERROR;
end case;

for COMMON_ACTIVITY_NUMBER in
    1 .. SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING_LIST_PACKAGE.
         LENGTH_OF
         (LIST => ACTIVITY_PTR.
                  LIST_OF_COMMON_ACTIVITIES)
loop

  OLD_CLAUSE(19 .. 39) := "o,o,o,o,o,o,o,o,o,o,o";

  case SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       SUBSTRING_OF
       (SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING_LIST_PACKAGE.
        ITEM_AT
```

```
     (LIST => ACTIVITY_PTR.
              LIST_OF_COMMON_ACTIVITIES,
      NODE_NUMBER => COMMON_ACTIVITY_NUMBER))(1) is
when 'E' => ESE_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            ASSERT
            (IN_CLAUSE => OLD_CLAUSE_QUERY,
            KB         => ESE_PERIOD_QUERY);

            -- ask

             ESE_PERIOD_INFERENCE_PACKAGE.
             SOLVE.
             START
             (THIS_QUERY => ESE_PERIOD_QUERY,
              THIS_KB    => ESE_PERIOD_KB);

             -- Get result

             ESE_PERIOD_INFERENCE_PACKAGE.
             CONTROL.
             GET_RESULT
             (OUT_LIST => VALUE_LIST);

             -- Release KB

             ESE_PERIOD_INFERENCE_PACKAGE.
             CONTROL.
             NO_MORE;

             -- Get the current weeks fom the response

            EXTRACT_WEEKS
            (VALUE_LIST => VALUE_LIST,
             OLD_CLAUSE => OLD_CLAUSE);

            -- Retract old clause

            ESE_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            RETRACT
            (CLAUSE => OLD_CLAUSE,
             KB     => ESE_PERIOD_KB);

            -- Assert cleared clause
            ESE_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            ASSERT
            (IN_CLAUSE   => NEW_CLAUSE,
             KB          => ESE_PERIOD_KB,
             AT_BACK_OF  => FALSE);

when 'I' => IT_PERIOD_INFERENCE_PACKAGE.
            LOGIC_KB.
            ASSERT
            (IN_CLAUSE => OLD_CLAUSE_QUERY,
             KB        => IT_PERIOD_QUERY);

             -- ask
```

```
                        IT_PERIOD_INFERENCE_PACKAGE.
                        SOLVE.
                        START
                        (THIS_QUERY => IT_PERIOD_QUERY,
                         THIS_KB    => IT_PERIOD_KB);

                        -- Get result

                        IT_PERIOD_INFERENCE_PACKAGE.
                        CONTROL.
                        GET_RESULT
                        (OUT_LIST => VALUE_LIST);

                        -- Release KB

                        IT_PERIOD_INFERENCE_PACKAGE.
                        CONTROL.
                        NO_MORE;

                        -- Get the current weeks fom the response

                        EXTRACT_WEEKS
                        (VALUE_LIST => VALUE_LIST,
                         OLD_CLAUSE => OLD_CLAUSE);

                        -- Retract old clause

                        IT_PERIOD_INFERENCE_PACKAGE.
                        LOGIC_KB.
                        RETRACT
                        (CLAUSE => OLD_CLAUSE,
                         KB     => IT_PERIOD_KB);

                        -- Assert cleared clause

                        IT_PERIOD_INFERENCE_PACKAGE.
                        LOGIC_KB.
                        ASSERT
                        (IN_CLAUSE  => NEW_CLAUSE,
                         KB         => IT_PERIOD_KB,
                         AT_BACK_OF => FALSE);

        when others => raise DEGREE_ERROR;
      end case;
    end loop;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR raised in PERIOD_4_PACKAGE " &
     "in  RESET");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_4_PACKAGE at " &
     "RESET");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_4_PACKAGE " &
     "at RESET");]
-------------------------------------------------------------------
```

```
end RESET;
-------------------------------------------------------------------

-------------------------------------------------------------------
procedure RESET_KNOWLEDGE_BASES
-------------------------------------------------------------------
(DAY          : in STANDARD.STRING;
 PERIOD       : in STANDARD.STRING;
 BLACKBOARD   : in TIMETABLE_BLACKBOARD_PACKAGE.
                   TIMETABLE_BLACKBOARD.
                   BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                 TIMETABLE_NODE_PTR_TYPE;

  PERIOD_NUMBER : TIMETABLE_TYPES_PACKAGE.
                  PERIOD_NUMBER_TYPE :=
                  TIMETABLE_TYPES_PACKAGE.
                  PERIOD_NUMBER_TYPE'
                  VALUE
                  (PERIOD);

  ITEM_INDEX : TIMETABLE_TYPES_PACKAGE.
               DAY_TYPE :=
               TIMETABLE_TYPES_PACKAGE.
               DAY_TYPE'
               VALUE
               (DAY);

  ACTIVITY_LIST : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_LIST_PACKAGE.
                  LIST_TYPE :=
                  TIMETABLE_BLACKBOARD_PACKAGE.
                  TIMETABLE_BLACKBOARD.
                  BLACKBOARD_ITEM
                  (BLACKBOARD  => BLACKBOARD,
                   LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                  DAYS,
                   ITEM_INDEX  => CONVERT(ITEM_INDEX)).
                  PERIOD(PERIOD_NUMBER).
                  SUPPORTERS;
-------------------------------------------------------------------
  PERIOD_ERROR : exception;
-------------------------------------------------------------------
begin

  for ACTIVITY_NUMBER in 1 .. TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_LIST_PACKAGE.
                              LENGTH_OF
                              (LIST => ACTIVITY_LIST)
  loop
    ACTIVITY_PTR := TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_LIST_PACKAGE.
                    ITEM_AT
                    (LIST        => ACTIVITY_LIST,
                     NODE_NUMBER => ACTIVITY_NUMBER);

    case ACTIVITY_PTR.NUMBER_OF_PERIODS is
```

```
      when 1 => RESET
                (DAY          => DAY,
                 PERIOD       => PERIOD,
                 ACTIVITY_PTR => ACTIVITY_PTR,
                 BLACKBOARD   => BLACKBOARD);

      when 2 => RESET
                (DAY          => DAY,
                 PERIOD       => TIMETABLE_TYPES_PACKAGE.
                                 PERIOD_NUMBER_TYPE'
                                 IMAGE
                                 (ACTIVITY_PTR.FIRST_PERIOD)
                                 (2 .. 2),
                 ACTIVITY_PTR => ACTIVITY_PTR,
                 BLACKBOARD   => BLACKBOARD);

                RESET
                (DAY          => DAY,
                 PERIOD       => TIMETABLE_TYPES_PACKAGE.
                                 PERIOD_NUMBER_TYPE'
                                 IMAGE
                                 (ACTIVITY_PTR.FIRST_PERIOD + 1)
                                 (2 .. 2),
                 ACTIVITY_PTR => ACTIVITY_PTR,
                 BLACKBOARD   => BLACKBOARD);

      when 3 => RESET
                (DAY          => DAY,
                 PERIOD       => TIMETABLE_TYPES_PACKAGE.
                                 PERIOD_NUMBER_TYPE'
                                 IMAGE
                                 (ACTIVITY_PTR.FIRST_PERIOD)
                                 (2 .. 2),
                 ACTIVITY_PTR => ACTIVITY_PTR,
                 BLACKBOARD   => BLACKBOARD);

                RESET
                (DAY          => DAY,
                 PERIOD       => TIMETABLE_TYPES_PACKAGE.
                                 PERIOD_NUMBER_TYPE'
                                 IMAGE
                                 (ACTIVITY_PTR.FIRST_PERIOD + 1)
                                 (2 .. 2),
                 ACTIVITY_PTR => ACTIVITY_PTR,
                 BLACKBOARD   => BLACKBOARD);

                RESET
                (DAY          => DAY,
                 PERIOD       => TIMETABLE_TYPES_PACKAGE.
                                 PERIOD_NUMBER_TYPE'
                                 IMAGE
                                 (ACTIVITY_PTR.FIRST_PERIOD + 2)
                                 (2 .. 2),
                 ACTIVITY_PTR => ACTIVITY_PTR,
                 BLACKBOARD   => BLACKBOARD);

      when others => raise PERIOD_ERROR;

   end case;
```

```
    end loop;
  ------------------------------------------------------------------------
  exception
    when PERIOD_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception PERIOD_ERROR at RESET_KNOWLEDGE_BASES " &
       "in PERIOD_4_PACKAGE");
  ------------------------------------------------------------------------
  end RESET_KNOWLEDGE_BASES;
  ------------------------------------------------------------------------

------------------------------------------------------------------------
end PERIOD_4_PACKAGE;
------------------------------------------------------------------------
```

Room KS

```
-------------------------------------------------------------------------
--
-- Unit      : ROOM_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 25 June 1993
-- Function : This package provides the room allocation operations
--
-------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE;
use  PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE;
-------------------------------------------------------------------------
package ROOM_PACKAGE is
-------------------------------------------------------------------------

  procedure RE_ALLOCATE_ROOM(
    PERIOD_PTR    : TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_NODE_PTR_TYPE;
    ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_NODE_PTR_TYPE;
    DAY           : TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_ITEM_TYPE;
    PERIOD        : TIMETABLE_TYPES_PACKAGE.
                    PERIOD_NUMBER_TYPE);

  procedure ALLOCATE_ROOMS
  (ROOMS           :  in out SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_ARRAY;
   PERIODS         :  in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_ARRAY;
   THIS_DAY        :  in     TIMETABLE_TYPES_PACKAGE.
                             DAY_TYPE;
   WEEKS_REQUIRED :  in     TIMETABLE_TYPES_PACKAGE.
                             WEEK_ARRAY_TYPE;
   ACTIVITY_PTR    :  in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
   SELECTED        :     out BOOLEAN);

-------------------------------------------------------------------------
end ROOM_PACKAGE;
-------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : ROOM_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield Universioty, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides room allocation
--
-------------------------------------------------------------------
package body ROOM_PACKAGE is
-------------------------------------------------------------------

  use TEXT_IO;

  package INTEGER_TEXT_IO is new INTEGER_IO(INTEGER);

  package ROOM_INFERENCE_PACKAGE is new
          LOGIC_INFERENCE_PACKAGE("ROOM");

  ROOM_KB : ROOM_INFERENCE_PACKAGE.
            LOGIC_KB.
            KB_RECORD;

  -----------------------------------------------------------------
  function RELEASE_ROOM_QUERY_STRING(
  -----------------------------------------------------------------
    ROOM   : STANDARD.STRING;
    DAY    : STANDARD.STRING;
    PERIOD : STANDARD.STRING) return STANDARD.STRING is
  -----------------------------------------------------------------
  begin

    return  "room(" &
             ROOM    & ",_," &
             DAY     & ","   &
             PERIOD  & ","   &
             "w(A,B,C,D,E,F,G,H,I,J,K)).";

  -----------------------------------------------------------------
  end RELEASE_ROOM_QUERY_STRING;
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  function SET_CLAUSE_STRING(
  -----------------------------------------------------------------
    ROOM   : STANDARD.STRING;
    SIZE   : STANDARD.STRING;
    DAY    : STANDARD.STRING;
    PERIOD : STANDARD.STRING) return STANDARD.STRING is
  -----------------------------------------------------------------
  begin

    return  "room("  &
             ROOM    & ","   &
             SIZE    & ","   &
             DAY     & ","   &
             PERIOD  & ","   &
             "w(A,B,C,D,E,F,G,H,I,J,K)).";
```

```
-------------------------------------------------------------------
end SET_CLAUSE_STRING;
-------------------------------------------------------------------
-------------------------------------------------------------------
procedure RELEASE_ROOM(
-------------------------------------------------------------------
  THIS_ROOM      : in SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      STRING;
  THIS_DAY       : in TIMETABLE_TYPES_PACKAGE.
                      DAY_TYPE;
  THIS_PERIOD    : in TIMETABLE_TYPES_PACKAGE.
                      PERIOD_NUMBER_TYPE;
  WEEKS_REQUIRED : in TIMETABLE_TYPES_PACKAGE.
                      WEEK_ARRAY_TYPE) is
-------------------------------------------------------------------

  ROOM : STANDARD.
         STRING(1 .. SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     LENGTH_OF
                     (THE_STRING => THIS_ROOM)) :=
                      SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      SUBSTRING_OF
                      (THE_STRING => THIS_ROOM);

  DAY : STANDARD.
        STRING(1 .. 3) :=  SYSTEM_TYPES_PACKAGE.
                           CONVERT
                           (TIMETABLE_TYPES_PACKAGE.
                            DAY_TYPE'
                            IMAGE
                            (THIS_DAY));

  PERIOD : STANDARD.
           STRING(1 .. 2) := "p" &
                             TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE'
                             IMAGE
                             (THIS_PERIOD)
                             (2 .. TIMETABLE_TYPES_PACKAGE.
                                   PERIOD_NUMBER_TYPE'
                                   IMAGE
                                   (THIS_PERIOD)'LAST);

  QUERY_STRING : STANDARD.
                 STRING(1 .. RELEASE_ROOM_QUERY_STRING
                             (ROOM   => ROOM,
                              DAY    => DAY,
                              PERIOD => PERIOD)'LENGTH) :=
                 RELEASE_ROOM_QUERY_STRING
                 (ROOM   => ROOM,
                  DAY    => DAY,
                  PERIOD => PERIOD);

  ROOM_SIZE  : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               STRING;
```

```
   QUERY_KB    : ROOM_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 KB_RECORD;

   VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING_LIST_PACKAGE.
                 LIST_TYPE;

-------------------------------------------------------------------
begin

   -- Assert query

   ROOM_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (KB         => QUERY_KB,
    IN_CLAUSE => QUERY_STRING);

   -- Ask

   ROOM_INFERENCE_PACKAGE.
   SOLVE.
   START
   (THIS_QUERY => QUERY_KB,
    THIS_KB    => ROOM_KB);

   -- Get result

   ROOM_INFERENCE_PACKAGE.
   CONTROL.
   GET_RESULT
   (OUT_LIST => VALUE_LIST);

   -- Release KB

   ROOM_INFERENCE_PACKAGE.
   CONTROL.
   NO_MORE;

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING_LIST_PACKAGE.
   DELETE_FRONT_OF
   (LIST => VALUE_LIST);

   -- Get the room size value

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING_LIST_PACKAGE.
   GET_FROM_FRONT_OF
   (LIST      => VALUE_LIST,
    THIS_ITEM => ROOM_SIZE);

   declare

     OLD_CLAUSE : STANDARD.STRING
                  (1 .. SET_CLAUSE_STRING
                        (ROOM    => ROOM,
                         SIZE    => SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
```

```
                                    SUBSTRING_OF
                                    (THE_STRING => ROOM_SIZE),
                                     DAY        => DAY,
                                     PERIOD     => PERIOD)'
                                     LENGTH) :=
                    SET_CLAUSE_STRING
                    (ROOM    => ROOM,
                     SIZE    => SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                SUBSTRING_OF
                                (THE_STRING => ROOM_SIZE),
                                 DAY        => DAY,
                                 PERIOD     => PERIOD);

  NEW_CLAUSE : STANDARD.STRING(1 .. OLD_CLAUSE'LENGTH) :=
               OLD_CLAUSE;

  OFFSET     : POSITIVE := OLD_CLAUSE'LENGTH - 24;

  INDEX      : POSITIVE := OFFSET + 1;

begin

--  Set all weeks in old and new clause strings

for WEEK in 1 .. SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING_LIST_PACKAGE.
                 LENGTH_OF
                 (LIST => VALUE_LIST) / 2
loop
  declare
    VALUE : SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            STRING;
  begin
    -- Delete varaiable name

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    DELETE_FRONT_OF
    (LIST => VALUE_LIST);

    -- Get the value

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => VALUE);

    -- Enter into old and new clauses

    OLD_CLAUSE(INDEX) := SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.
                         SUBSTRING_OF
                         (THE_STRING => VALUE)(1);

    NEW_CLAUSE(INDEX) := SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.
                         SUBSTRING_OF
```

```
                                (THE_STRING => VALUE)(1);

        INDEX := INDEX + 2;

      end;
    end loop;

    -- Adjust new clause for weeks not now required

    for WEEK in TIMETABLE_TYPES_PACKAGE.
                WEEK_NUMBER_TYPE
    loop
      if WEEKS_REQUIRED(WEEK) = TRUE then
        NEW_CLAUSE(WEEK + OFFSET) := 'o';
      end if;
      OFFSET := OFFSET + 1;
    end loop;

    -- Retract old clause

    ROOM_INFERENCE_PACKAGE.
    LOGIC_KB.
    RETRACT
    (CLAUSE => OLD_CLAUSE,
     KB     => ROOM_KB);

    -- Assert new clause

    ROOM_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE => NEW_CLAUSE,
     KB        => ROOM_KB);

    end;
  ------------------------------------------------------------------
  end RELEASE_ROOM;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure RE_ALLOCATE_ROOM(
  ------------------------------------------------------------------
    PERIOD_PTR   : TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
    ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
    DAY          : TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_ITEM_TYPE;
    PERIOD       : TIMETABLE_TYPES_PACKAGE.
                   PERIOD_NUMBER_TYPE) is
  ------------------------------------------------------------------

    RESOURCE : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               STRING;

    ------------------------------------------------------------------
    function CONVERT
```

```
  ------------------------------------------------------------------
  (DAY : in TIMETABLE_TYPES_PACKAGE.
            DAY_TYPE) return  TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_ITEM_TYPE  is
  ------------------------------------------------------------------
  DAY_ERROR : exception;
  ------------------------------------------------------------------
  begin

    case DAY is

    when TIMETABLE_TYPES_PACKAGE.MON =>
         return TIMETABLE_TYPES_PACKAGE.MONDAY;
    when TIMETABLE_TYPES_PACKAGE.TUE =>
         return TIMETABLE_TYPES_PACKAGE.TUESDAY;
    when TIMETABLE_TYPES_PACKAGE.WED =>
         return TIMETABLE_TYPES_PACKAGE.WEDNESDAY;
    when TIMETABLE_TYPES_PACKAGE.THU =>
         return TIMETABLE_TYPES_PACKAGE.THURSDAY;
    when TIMETABLE_TYPES_PACKAGE.FRI =>
         return TIMETABLE_TYPES_PACKAGE.FRIDAY;
    when others                      =>
         raise DAY_ERROR;

    end case;
  ------------------------------------------------------------------
  exception
    when DAY_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception DAY_ERROR in ROOM_KS_PACKAGE at " &
       "CONVERT");
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in ROOM_KS_PACKAGE at " &
       "CONVERT");
  ------------------------------------------------------------------
  end CONVERT;
  ------------------------------------------------------------------

----------------------------------------------------------------------
begin

  -- Set the slot name

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "RESOURCE",
   TO_THE_STRING      => RESOURCE);

  -- Re_allocate room for each staff member

  for STAFF_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           LENGTH_OF
                           (LIST => ACTIVITY_PTR.
                                    STAFF_LIST)
  loop
    declare
```

```
   STAFF_NAME      : SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     STRING;
   FACET_PTR       : TIMETABLE_TYPES_PACKAGE.
                     PERIOD_FACET_RECORD_PTR_TYPE;
   WEEKS_REQUIRED  : TIMETABLE_TYPES_PACKAGE.
                     WEEK_ARRAY_TYPE;
   ROOM            : SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING_ARRAY(1..1);
   PERIOD_ARRAY    : SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING_ARRAY(1..1);
   SELECTED        : BOOLEAN := FALSE;

begin

   -- Get staff name

   SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   COPY
   (FROM_THE_STRING => SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING_LIST_PACKAGE.
                       ITEM_AT
                       (LIST        => ACTIVITY_PTR.
                                       STAFF_LIST,
                        NODE_NUMBER => STAFF_NUMBER),
    TO_THE_STRING   => STAFF_NAME);

   -- Get the facet pointer for this staff member

   TIMETABLE_TYPES_PACKAGE.
   PERIOD_FRAME_BASE_PACKAGE.
   FIND_FACET
   (FACET_NAME         => STAFF_NAME,
    SLOT_NAME          => RESOURCE,
    FRAME_NAME         => ACTIVITY_PTR.
                          FACT,
    FRAME_BASE_RECORD => PERIOD_PTR.
                          PERIOD_DETAIL_FRAMES,
    FACET_PTR          => FACET_PTR);

   -- Get the weeks this staff member is involved

   PERIOD_2_PACKAGE.
   CONVERT
   (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING.
                  SUBSTRING_OF
                  (THE_STRING => FACET_PTR.
                                 WEEK_CODE),
    WEEK_ARRAY => WEEKS_REQUIRED);

   -- Release the room

   RELEASE_ROOM
   (THIS_ROOM       => FACET_PTR.
                       ROOM,
    THIS_DAY        => CONVERT
                       (DAY => DAY),
    THIS_PERIOD     => PERIOD,
```

```
        WEEKS_REQUIRED => WEEKS_REQUIRED);

      -- Clear the room allocation from facet

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      CLEAR
      (THE_STRING => FACET_PTR.
                     ROOM);

       -- Allocate a new room

       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       COPY
       (FROM_THE_SUBSTRING => "p",
        TO_THE_STRING      => PERIOD_ARRAY(1));

       SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       APPEND
       (TO_THE_STRING => PERIOD_ARRAY(1),
        THE_SUBSTRING => TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE'
                         IMAGE
                         (PERIOD)(2 .. TIMETABLE_TYPES_PACKAGE.
                                       PERIOD_NUMBER_TYPE'
                                       IMAGE
                                       (PERIOD)'LAST));

       ALLOCATE_ROOMS
       (ROOMS           => ROOM,
        PERIODS         => PERIOD_ARRAY,
        THIS_DAY        => CONVERT
                           (DAY => DAY),
        WEEKS_REQUIRED => WEEKS_REQUIRED,
        ACTIVITY_PTR    => ACTIVITY_PTR,
        SELECTED        => SELECTED);

      -- Update room in facet

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_STRING => ROOM(1),
       TO_THE_STRING   => FACET_PTR.ROOM);

    end;

  end loop;

-------------------------------------------------------------------
end RE_ALLOCATE_ROOM;
-------------------------------------------------------------------

-------------------------------------------------------------------
function QUERY_STRING
-------------------------------------------------------------------
(NUMBER_OF_PERIODS  : in POSITIVE;
 PERIODS            : in SYSTEM_TYPES_PACKAGE.
```

```
                              DYNAMIC_STRING_ARRAY;
 STUDENTS              : in STANDARD.STRING;
 DAY                   : in STANDARD.STRING)
return STANDARD.STRING is
-------------------------------------------------------------------------
 PERIOD_ERROR : exception;
 P1           : STANDARD.STRING(1 .. 2);
 P2           : STANDARD.STRING(1 .. 2);
 P3           : STANDARD.STRING(1 .. 2);
 INDEX        : STANDARD.POSITIVE;
-------------------------------------------------------------------------
begin
  INDEX := PERIODS'FIRST;
  case NUMBER_OF_PERIODS is
    when 1 => P1 := SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => PERIODS(INDEX));
    when 2 => P1 := SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => PERIODS(INDEX));
              P2 := SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => PERIODS(INDEX + 1));
    when 3 => P1 := SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => PERIODS(INDEX));
              P2 := SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => PERIODS(INDEX + 1));
              P3 := SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => PERIODS(INDEX + 2));
    when others => raise PERIOD_ERROR;
  end case;

  case NUMBER_OF_PERIODS is
    when 1 => return
              "room_periods_1" &
              "(R"        & "," &
                STUDENTS  & "," &
                DAY       & "," &
                P1        & "," &
               "w(A,B,C,D,E,F,G,H,I,J,K)).";
    when 2 => return
              "room_periods_2" &
              "(R"        & "," &
                STUDENTS  & "," &
                DAY       & "," &
                P1        & "," &
                P2        & "," &
               "w(A,B,C,D,E,F,G,H,I,J,K)).";
    when 3 => return
              "room_periods_3" &
              "(R"        & "," &
```

```
                STUDENTS & "," &
                DAY      & "," &
                P1       & "," &
                P2       & "," &
                P3       & "," &
                "w(A,B,C,D,E,F,G,H,I,J,K)).";
    when others => raise PERIOD_ERROR;
  end case;
-------------------------------------------------------------------
exception
  when PERIOD_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception PERIOD_ERROR raised at ROOM_QUERY_STRING " &
     "in ROOM_PACKAGE");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR raised at ROOM_QUERY_STRING " &
     "in ROOM_PACKAGE");
-------------------------------------------------------------------
end QUERY_STRING;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure EXTRACT_ROOM
-------------------------------------------------------------------
(ROOM               : in out SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING.
                             STRING;
 NUMBER_OF_PERIODS  : in     POSITIVE := 1;
 PERIODS            : in     SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_ARRAY;
 DAY                : in     STANDARD.STRING;
 VALUE_LIST         : in out SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_LIST_PACKAGE.
                             LIST_TYPE;
 ROOM_KB            : in out ROOM_INFERENCE_PACKAGE.
                             LOGIC_KB.
                             KB_RECORD) is
-------------------------------------------------------------------
  PERIOD_ERROR : exception;

  OLD_CLAUSES : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_ARRAY
                (PERIODS'FIRST .. PERIODS'LAST);
  NEW_CLAUSES : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_ARRAY
                (PERIODS'FIRST .. PERIODS'LAST);
  TOKEN       : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                STRING;
-------------------------------------------------------------------
begin

  -- Pop variable name R

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING_LIST_PACKAGE.
  DELETE_FRONT_OF
  (LIST => VALUE_LIST);
```

```
-- Get the room name eg mh121

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING_LIST_PACKAGE.
GET_FROM_FRONT_OF
(LIST      => VALUE_LIST,
 THIS_ITEM => TOKEN);

-- Record room for return

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_STRING => TOKEN,
 TO_THE_STRING   => ROOM);

  -- Get the room capacity

  declare

    CAPACITY_QUERY_STRING : SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            STRING;
    CAPACITY_RESPONSE     : SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE;
    CAPACITY_QUERY        : ROOM_INFERENCE_PACKAGE.
                            LOGIC_KB.
                            KB_RECORD;

  begin

    -- Build query

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    COPY
    (FROM_THE_STRING => TOKEN,
     TO_THE_STRING   => CAPACITY_QUERY_STRING);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (TO_THE_STRING => CAPACITY_QUERY_STRING,
     THE_SUBSTRING => "(C).");

    -- Assert the query

    ROOM_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE => SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING.
                  SUBSTRING_OF
                  (THE_STRING => CAPACITY_QUERY_STRING),
     KB        => CAPACITY_QUERY);

    -- Ask the query
```

```
    ROOM_INFERENCE_PACKAGE.
    SOLVE.
    START
    (THIS_QUERY => CAPACITY_QUERY,
     THIS_KB    => ROOM_KB);

    ROOM_INFERENCE_PACKAGE.
    CONTROL.
    GET_RESULT
    (OUT_LIST => CAPACITY_RESPONSE);

    ROOM_INFERENCE_PACKAGE.
    CONTROL.
    NO_MORE;

for PERIOD in PERIODS'FIRST .. PERIODS'LAST
loop

  -- Add "room("

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (TO_THE_STRING      => OLD_CLAUSES(PERIOD),
   FROM_THE_SUBSTRING => "room(");

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (TO_THE_STRING => NEW_CLAUSES(PERIOD),
   FROM_THE_SUBSTRING => "room(");

end loop;

  -- Add room name

for PERIOD in PERIODS'FIRST .. PERIODS'LAST
loop

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  APPEND
  (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => TOKEN) & ",",
   TO_THE_STRING => OLD_CLAUSES(PERIOD));

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  APPEND
  (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => TOKEN) & ",",
   TO_THE_STRING => NEW_CLAUSES(PERIOD));

end loop;

    -- Add capacity to query string
```

```
for PERIOD in PERIODS'FIRST .. PERIODS'LAST
loop
    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (TO_THE_STRING   => OLD_CLAUSES(PERIOD),
     THE_STRING      => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING_LIST_PACKAGE.
                        ITEM_AT
                        (LIST        => CAPACITY_RESPONSE,
                         NODE_NUMBER => 2));

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (TO_THE_STRING   => NEW_CLAUSES(PERIOD),
     THE_STRING      => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING_LIST_PACKAGE.
                        ITEM_AT
                        (LIST        => CAPACITY_RESPONSE,
                         NODE_NUMBER => 2));

end loop;

  -- Add day and period

for PERIOD in PERIODS'FIRST .. PERIODS'LAST
loop

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  APPEND
  (THE_SUBSTRING => "," & DAY & ",",
   TO_THE_STRING => OLD_CLAUSES(PERIOD));

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (THE_SUBSTRING => "," & DAY & ",",
     TO_THE_STRING => NEW_CLAUSES(PERIOD));

end loop;

for PERIOD in PERIODS'FIRST .. PERIODS'LAST
loop

    -- Add the period

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      SUBSTRING_OF
                      (THE_STRING => PERIODS(PERIOD)),
     TO_THE_STRING => OLD_CLAUSES(PERIOD));

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
```

```
        APPEND
        (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          SUBSTRING_OF
                          (THE_STRING => PERIODS(PERIOD)),
         TO_THE_STRING => NEW_CLAUSES(PERIOD));

        -- Add ",w("

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        APPEND
        (THE_SUBSTRING => ",w(",
         TO_THE_STRING      => OLD_CLAUSES(PERIOD));

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        APPEND
        (THE_SUBSTRING => ",w(",
         TO_THE_STRING => NEW_CLAUSES(PERIOD));

    end loop;

    -- Add the weeks

    for WEEK in 'A' .. 'K'
    loop

        if not SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               IS_EMPTY
               (LIST => VALUE_LIST)

               and then

               SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               SUBSTRING_OF
               (THE_STRING => SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING_LIST_PACKAGE.
                              FRONT_OF
                              (LIST => VALUE_LIST))(1) = WEEK then

         -- Clear token

          if not SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 IS_NULL
                 (THE_STRING => TOKEN) then

            SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            CLEAR
            (THE_STRING => TOKEN);

          end if;

          -- Delete variable name

          SYSTEM_TYPES_PACKAGE.
```

```
        DYNAMIC_STRING_LIST_PACKAGE.
        DELETE_FRONT_OF
        (LIST => VALUE_LIST);

        -- Get the value

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING_LIST_PACKAGE.
        GET_FROM_FRONT_OF
        (LIST      => VALUE_LIST,
         THIS_ITEM => TOKEN);

        -- Add value

        for PERIOD in PERIODS'FIRST .. PERIODS'LAST
        loop

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        APPEND
        (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          SUBSTRING_OF
                          (THE_STRING => TOKEN),
         TO_THE_STRING => OLD_CLAUSES(PERIOD));

        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING.
        APPEND
        (THE_SUBSTRING => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          SUBSTRING_OF
                          (THE_STRING => TOKEN),
         TO_THE_STRING => NEW_CLAUSES(PERIOD));

        if WEEK /= 'K' then  -- Add ','

          SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
          APPEND
          (TO_THE_STRING => OLD_CLAUSES(PERIOD),
           THE_SUBSTRING => ",");

          SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
          APPEND
          (TO_THE_STRING => NEW_CLAUSES(PERIOD),
           THE_SUBSTRING => ",");

        end if;

        end loop;

      else

        for PERIOD in PERIODS'FIRST .. PERIODS'LAST
        loop

          SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
```

```
          APPEND
          ( TO_THE_STRING => OLD_CLAUSES(PERIOD),
            THE_SUBSTRING => "o");

          SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING.
          APPEND
          ( TO_THE_STRING => NEW_CLAUSES(PERIOD),
            THE_SUBSTRING => "x");

          if WEEK /= 'K' then  -- Add ','

            SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            APPEND
            (TO_THE_STRING => OLD_CLAUSES(PERIOD),
             THE_SUBSTRING => ",");

            SYSTEM_TYPES_PACKAGE.
            DYNAMIC_STRING.
            APPEND
            (TO_THE_STRING => NEW_CLAUSES(PERIOD),
             THE_SUBSTRING => ",");

          end if;

        end loop;

      end if;

    end loop;

  for PERIOD in PERIODS'FIRST .. PERIODS'LAST
  loop

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (TO_THE_STRING => OLD_CLAUSES(PERIOD),
     THE_SUBSTRING => ")).");

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING.
    APPEND
    (TO_THE_STRING => NEW_CLAUSES(PERIOD),
     THE_SUBSTRING => ")).");

  end loop;

  end;

  -- Retract all old clauses

  for PERIOD in PERIODS'FIRST .. PERIODS'LAST
  loop

   ROOM_INFERENCE_PACKAGE.
   LOGIC_KB.
   RETRACT
   (CLAUSE => SYSTEM_TYPES_PACKAGE.
```

```
                DYNAMIC_STRING.
                SUBSTRING_OF
                (THE_STRING => OLD_CLAUSES(PERIOD)),
     KB      => ROOM_KB);

    ROOM_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE  => SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   SUBSTRING_OF
                   (THE_STRING => NEW_CLAUSES(PERIOD)),
     KB         => ROOM_KB,
     AT_BACK_OF => FALSE);

  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR raised at EXTRACT_ROOMS " &
     "in ROOM_PACKAGE");
  when PERIOD_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception PERIOD_ERROR raised at EXTRACT_ROOMS " &
     "in ROOM_PACKAGE");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS raised at EXTRACT_ROOMS " &
     "in ROOM_PACKAGE");
-------------------------------------------------------------------
end EXTRACT_ROOM;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ALLOCATE_ROOMS
-------------------------------------------------------------------
(ROOMS           : in out SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_ARRAY;
 PERIODS         : in     SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_ARRAY;
 THIS_DAY        : in     TIMETABLE_TYPES_PACKAGE.
                          DAY_TYPE;
 WEEKS_REQUIRED  : in     TIMETABLE_TYPES_PACKAGE.
                          WEEK_ARRAY_TYPE;
 ACTIVITY_PTR    : in     TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_NODE_PTR_TYPE;
 SELECTED        :    out BOOLEAN) is
-------------------------------------------------------------------
  STUDENTS : STANDARD.STRING
             (1 .. STANDARD.POSITIVE'
                   IMAGE
                   (ACTIVITY_PTR.STUDENTS)'LENGTH - 1) :=
             STANDARD.POSITIVE'
             IMAGE
             (ACTIVITY_PTR.STUDENTS)
             (2 .. STANDARD.POSITIVE'
                   IMAGE
                   (ACTIVITY_PTR.STUDENTS)'LAST);
```

```
  DAY : STANDARD.STRING(1 .. 3) := SYSTEM_TYPES_PACKAGE.
                                   CONVERT
                                   (TIMETABLE_TYPES_PACKAGE.
                                    DAY_TYPE'
                                    IMAGE
                                    (THIS_DAY));

  NUMBER_OF_ROOMS    : STANDARD.NATURAL := ROOMS'LENGTH;

  NUMBER_OF_PERIODS : STANDARD.NATURAL := PERIODS'LENGTH;

  ROOMS_NOT_AVAILABLE : exception;
-------------------------------------------------------------------
begin

  for THIS_ROOM in ROOMS'FIRST .. ROOMS'LAST
  loop

    declare

      ROOM : SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING.
             STRING;

      ROOM_QUERY_STRING : STRING
                          (1 .. QUERY_STRING
                          (NUMBER_OF_PERIODS => NUMBER_OF_PERIODS,
                           PERIODS           => PERIODS,
                           STUDENTS          => STUDENTS,
                           DAY               => DAY)'LENGTH);

      ROOM_QUERY_OFFSET : POSITIVE :=
                          ROOM_QUERY_STRING'LENGTH - 24;

      ROOM_QUERY : ROOM_INFERENCE_PACKAGE.
                   LOGIC_KB.
                   KB_RECORD;

      VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING_LIST_PACKAGE.
                   LIST_TYPE;
    begin

     ROOM_QUERY_STRING := QUERY_STRING
                          (NUMBER_OF_PERIODS => NUMBER_OF_PERIODS,
                           PERIODS           => PERIODS,
                           STUDENTS          => STUDENTS,
                           DAY               => DAY);

      -- Set the query string where 'o' signifies week required

        for WEEK in TIMETABLE_TYPES_PACKAGE.
                    WEEK_NUMBER_TYPE
        loop

          if WEEKS_REQUIRED(WEEK) = TRUE then
            ROOM_QUERY_STRING(WEEK + ROOM_QUERY_OFFSET) := 'o';
          end if;
          ROOM_QUERY_OFFSET := ROOM_QUERY_OFFSET + 1;
```

```
  end loop;

-- Assert the query

ROOM_INFERENCE_PACKAGE.
LOGIC_KB.
ASSERT
(KB         => ROOM_QUERY,
 IN_CLAUSE => ROOM_QUERY_STRING);

-- Ask the query

ROOM_INFERENCE_PACKAGE.
SOLVE.
START
(THIS_QUERY => ROOM_QUERY,
 THIS_KB     => ROOM_KB);


ROOM_INFERENCE_PACKAGE.
CONTROL.
GET_RESULT
(OUT_LIST => VALUE_LIST);

-- Check for No periods

if SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   IS_EQUAL
   (LEFT  => "No",
    RIGHT => SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING_LIST_PACKAGE.
             ITEM_AT
             (LIST        => VALUE_LIST,
              NODE_NUMBER => 1)) then

  raise ROOMS_NOT_AVAILABLE;

end if;

-- Release solve

ROOM_INFERENCE_PACKAGE.
CONTROL.
NO_MORE;

-- Extract rooms

EXTRACT_ROOM
(ROOM              => ROOM,
 NUMBER_OF_PERIODS => NUMBER_OF_PERIODS,
 PERIODS           => PERIODS,
 DAY               => DAY,
 VALUE_LIST        => VALUE_LIST,
 ROOM_KB           => ROOM_KB);

-- Add room to room array for return

SYSTEM_TYPES_PACKAGE.
```

```
        DYNAMIC_STRING.
        COPY
        (FROM_THE_STRING => ROOM,
         TO_THE_STRING   => ROOMS(THIS_ROOM));

      end;

    end loop;

    SELECTED := TRUE;
  ------------------------------------------------------------------
  exception
    when ROOMS_NOT_AVAILABLE =>
      SELECTED := FALSE;
      TEXT_IO.PUT_LINE
      ("Exception ROOMS_NOT_AVALABLE in ROOM_PACKAGE " &
       "at ALLOCATE_ROOMS");
    when CONSTRAINT_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception CONSTRAINT_ERROR in ROOM_PACKAGE " &
       "at ALLOCATE_ROOMS ");
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in ROOM_PACKAGE " &
       "at ALLOCATE_ROOMS ");
  ------------------------------------------------------------------
  end ALLOCATE_ROOMS;
  ------------------------------------------------------------------
begin

  ROOM_INFERENCE_PACKAGE.
  LOGIC_KB.
  BUILD
  (KB        => ROOM_KB,
   FILE_NAME => "room.pro");

------------------------------------------------------------------
end ROOM_PACKAGE;
------------------------------------------------------------------
```

Period ESE

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_ESE_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides the blackboard transformation
--             from the degree activities level to the day level by
--             alllocating periods to the degree activities
--
-------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE;
use  PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE;
-------------------------------------------------------------------
package PERIOD_ESE_PACKAGE is
-------------------------------------------------------------------

  function PREFERED_ESE_PERIOD_OCCUPIED
  (WEEKS_REQUIRED : in TIMETABLE_TYPES_PACKAGE.
                       WEEK_ARRAY_TYPE;
   ESE_PERIOD_KB  : in ESE_PERIOD_INFERENCE_PACKAGE.
                       LOGIC_KB.
                       KB_RECORD;
   ACTIVITY_PTR   : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE
  ) return BOOLEAN;

  procedure RESERVE_1_PREFERED_ESE_PERIOD
  (PERIOD          : in     STANDARD.STRING;
   THIS_PERIOD     : in out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
   DAY             : in     STANDARD.STRING;
   THIS_DAY        : in out TIMETABLE_TYPES_PACKAGE.
                            DAY_TYPE;
   WEEKS_REQUIRED  : in     TIMETABLE_TYPES_PACKAGE.
                            WEEK_ARRAY_TYPE;
   PERIOD_KB       : in out ESE_PERIOD_INFERENCE_PACKAGE.
                            LOGIC_KB.
                            KB_RECORD;
   ACTIVITY_PTR    : in     TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD      : in out TIMETABLE_BLACKBOARD_PACKAGE.
                            TIMETABLE_BLACKBOARD.
                            BLACKBOARD_TYPE;
```

```
   SELECTED        :     out BOOLEAN);

  procedure SELECT_1_ESE_PERIOD
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
   PERIOD_KB       :  in out ESE_PERIOD_INFERENCE_PACKAGE.
                           LOGIC_KB.
                           KB_RECORD;
   ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
   SELECTED        :     out BOOLEAN);

  procedure SELECT_2_ESE_PERIODS
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
   PERIOD_KB       :  in out ESE_PERIOD_INFERENCE_PACKAGE.
                           LOGIC_KB.
                           KB_RECORD;
   ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
   SELECTED        :     out BOOLEAN);

  procedure SELECT_3_ESE_PERIODS
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   PERIOD_3        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
   PERIOD_KB       :  in out ESE_PERIOD_INFERENCE_PACKAGE.
                           LOGIC_KB.
                           KB_RECORD;
   ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
   SELECTED        :     out BOOLEAN);

-------------------------------------------------------------------
end PERIOD_ESE_PACKAGE;
-------------------------------------------------------------------
```

```
-----------------------------------------------------------------------
--
-- Unit     : PERIOD_ESE_PACKAGE body
-- Author   : A Harrison, Software Engineering Group,
--            Cranfield University, RMCS, Shrivenham
-- Date     : 29 July 1992
-- Function : This package provides the blackboard transformation
--            from the
--            degree activity level to the day level by allocating
--            periods to the degree activities
--
-----------------------------------------------------------------------
package body PERIOD_ESE_PACKAGE is
-----------------------------------------------------------------------
  use TEXT_IO;
-----------------------------------------------------------------------
  function PREFERED_ESE_PERIOD_OCCUPIED
-----------------------------------------------------------------------
  (WEEKS_REQUIRED : in TIMETABLE_TYPES_PACKAGE.
                       WEEK_ARRAY_TYPE;
   ESE_PERIOD_KB  : in ESE_PERIOD_INFERENCE_PACKAGE.
                       LOGIC_KB.
                       KB_RECORD;
   ACTIVITY_PTR   : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE
  ) return BOOLEAN is
-----------------------------------------------------------------------
    PERIOD_QUERY_STRING : STANDARD.STRING(1 ..42) :=
                          "period(" &
                          SYSTEM_TYPES_PACKAGE.
                          CONVERT
                          (SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           SUBSTRING_OF
                           (THE_STRING => ACTIVITY_PTR.
                                          DAY_PREFERENCE)) &
                          ",p" &
                          SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          SUBSTRING_OF
                          (THE_STRING => ACTIVITY_PTR.
                                         PERIOD_PREFERENCE) &
                          ",wks(_,_,_,_,_,_,_,_,_,_)).";

    PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                   LOGIC_KB.
                   KB_RECORD;

    VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING_LIST_PACKAGE.
                 LIST_TYPE;

-----------------------------------------------------------------------
  begin

    -- Set the query

    declare
      PERIOD_QUERY_OFFSET : POSITIVE := 18;
    begin
```

```
    for WEEK in TIMETABLE_TYPES_PACKAGE.
                WEEK_NUMBER_TYPE
    loop
      if WEEKS_REQUIRED(WEEK) then
        PERIOD_QUERY_STRING(WEEK + PERIOD_QUERY_OFFSET) := 'o';
      end if;
      PERIOD_QUERY_OFFSET := PERIOD_QUERY_OFFSET + 1;
    end loop;

    -- Set query

    ESE_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (KB         => PERIOD_QUERY,
     IN_CLAUSE => PERIOD_QUERY_STRING);

    -- Ask the query

    ESE_PERIOD_INFERENCE_PACKAGE.
    SOLVE.
    START
    (THIS_QUERY => PERIOD_QUERY,
     THIS_KB    => ESE_PERIOD_KB);

    -- Get the result

    ESE_PERIOD_INFERENCE_PACKAGE.
    CONTROL.
    GET_RESULT
    (OUT_LIST => VALUE_LIST);

   -- Check for 'No'

   if SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      IS_EQUAL
      (LEFT  => "No",
       RIGHT => SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_LIST_PACKAGE.
                ITEM_AT
                (LIST        => VALUE_LIST,
                 NODE_NUMBER => 1)) then

     return TRUE;

   else

     -- Release control

     ESE_PERIOD_INFERENCE_PACKAGE.
     CONTROL.
     NO_MORE;

     return FALSE;

   end if;
  end;
-------------------------------------------------------------------
exception
```

```
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_KS_PACKAGE at " &
     "PREFERED_ESE_PERIOD_OCCUPIED");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_KS_PACKAGE " &
     "at PREFERED_ESE_PERIOD_OCCUPIED");
-------------------------------------------------------------------
end PREFERED_ESE_PERIOD_OCCUPIED;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure RESERVE_1_PREFERED_ESE_PERIOD
-------------------------------------------------------------------
(PERIOD          : in     STANDARD.STRING;
 THIS_PERIOD     : in out TIMETABLE_TYPES_PACKAGE.
                          PERIOD_NUMBER_TYPE;
 DAY             : in     STANDARD.STRING;
 THIS_DAY        : in out TIMETABLE_TYPES_PACKAGE.
                          DAY_TYPE;
 WEEKS_REQUIRED  : in     TIMETABLE_TYPES_PACKAGE.
                          WEEK_ARRAY_TYPE;
 PERIOD_KB       : in out ESE_PERIOD_INFERENCE_PACKAGE.
                          LOGIC_KB.
                          KB_RECORD;
 ACTIVITY_PTR    : in     TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_NODE_PTR_TYPE;
 BLACKBOARD      : in out TIMETABLE_BLACKBOARD_PACKAGE.
                          TIMETABLE_BLACKBOARD.
                          BLACKBOARD_TYPE;
 SELECTED        :    out BOOLEAN) is
-------------------------------------------------------------------

  OLD_CLAUSE : STANDARD.STRING (1 .. 42) :=
               "period(" &
               SYSTEM_TYPES_PACKAGE.
               CONVERT(DAY) &
               ",p" &
               PERIOD &
               ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

  NEW_CLAUSE : STANDARD.STRING (1 .. 42) :=
               "period(" &
               SYSTEM_TYPES_PACKAGE.
               CONVERT(DAY) &
               ",p" &
               PERIOD &
               ",wks(o,o,o,o,o,o,o,o,o,o,o)).";

  PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;
-------------------------------------------------------------------
begin
```

```
SELECTED := TRUE;

THIS_PERIOD := TIMETABLE_TYPES_PACKAGE.
               PERIOD_NUMBER_TYPE'
               VALUE
               (PERIOD);

THIS_DAY := TIMETABLE_TYPES_PACKAGE.
            DAY_TYPE'
            VALUE
            (DAY);

declare
  PERIOD_OFFSET : POSITIVE := 18;
begin
  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop
    if WEEKS_REQUIRED(WEEK) then
      NEW_CLAUSE(WEEK + PERIOD_OFFSET) := 'x';
    end if;
    PERIOD_OFFSET := PERIOD_OFFSET + 1;
  end loop;
end;

-- Find old clause

ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
ASSERT
(KB => PERIOD_QUERY,
 IN_CLAUSE => OLD_CLAUSE);

ESE_PERIOD_INFERENCE_PACKAGE.
SOLVE.
START
(THIS_QUERY => PERIOD_QUERY,
 THIS_KB    => PERIOD_KB);

ESE_PERIOD_INFERENCE_PACKAGE.
CONTROL.
GET_RESULT
(OUT_LIST => VALUE_LIST);

EXTRACT_WEEKS
(VALUE_LIST => VALUE_LIST,
 OLD_CLAUSE => OLD_CLAUSE);

ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_CLAUSE,
 KB     => ESE_PERIOD_KB);

ESE_PERIOD_INFERENCE_PACKAGE.
CONTROL.
NO_MORE;

-- Assert new clause
```

```
  if NEW_CLAUSE(19 .. 39) = "x,x,x,x,x,x,x,x,x,x,x" then
    ESE_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE => NEW_CLAUSE,
     KB        => ESE_PERIOD_KB);
  else
    ESE_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE  => NEW_CLAUSE,
     KB         => ESE_PERIOD_KB,
     AT_BACK_OF => FALSE);
  end if;

  -- Do same for all common modules

  ADJUST_COMMON_MODULE_PERIODS
  (DAY            => SYSTEM_TYPES_PACKAGE.CONVERT(DAY),
   PERIOD         => "p" &.PERIOD,
   WEEKS_REQUIRED => WEEKS_REQUIRED,
   ACTIVITY_PTR   => ACTIVITY_PTR);

  -- Add periods to blackboard

  ADD_PERIOD_DETAILS
  (THIS_DAY       => THIS_DAY,
   FIRST_PERIOD   => THIS_PERIOD,
   LAST_PERIOD    => THIS_PERIOD,
   WEEKS_REQUIRED => WEEKS_REQUIRED,
   ACTIVITY_PTR   => ACTIVITY_PTR,
   BLACKBOARD     => BLACKBOARD);

  -- Make staff busy

  STAFF_PACKAGE.
  MAKE_ALL_STAFF_BUSY
  (STAFF_LIST              => ACTIVITY_PTR.
                              STAFF_LIST,
  MODULE                   => ACTIVITY_PTR.
                              FACT,
  DAY                      => THIS_DAY,
  PERIOD                   => THIS_PERIOD,
  WEEK_ARRAY               => WEEKS_REQUIRED,
  STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                              STAFF_FRAME_BASE_RECORD);
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in PERIOD_ESE_PACKAGE at " &
     "RESERVE_1_PREFERED_ESE_PERIOD");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_ESE_PACKAGE " &
     "at RESERVE_1_PREFERED_ESE_PERIOD");
-------------------------------------------------------------------
end RESERVE_1_PREFERED_ESE_PERIOD;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Select period identifies a period/day that satisfies the
-- current requirement
--
-------------------------------------------------------------------
procedure SELECT_1_ESE_PERIOD
-------------------------------------------------------------------
(PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                         DAY_TYPE;
 WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                         WEEK_ARRAY_TYPE;
 PERIOD_KB       :  in out ESE_PERIOD_INFERENCE_PACKAGE.
                         LOGIC_KB.
                         KB_RECORD;
 ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
 SELECTED        :     out BOOLEAN) is
-------------------------------------------------------------------

  PERIOD_QUERY_STRING :
  STANDARD.
  STRING
  (1 .. 43) := "one_period(A,B,wks(C,D,E,F,G,H,I,J,K,L,M)).";

  COMMON_QUERY_STRING :
  STANDARD.
  STRING(1 .. 42) := "period(    ,   ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

  OLD_CLAUSE_STRING    :
  STANDARD.
  STRING(1 .. 42) :="period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

  NEW_CLAUSE_STRING    :
  STANDARD.
  STRING(1 .. 42) := "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

  PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;

  PERIODS_NOT_AVAILABLE : exception;

  PERIOD :  TIMETABLE_TYPES_PACKAGE.
            PERIOD_NUMBER_TYPE;
  DAY    :  TIMETABLE_TYPES_PACKAGE.
            DAY_TYPE;

-------------------------------------------------------------------
begin

  -- Set the query string where 'o' signifies week required
```

```
declare
  PERIOD_QUERY_OFFSET : POSITIVE := 19;
begin

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      PERIOD_QUERY_STRING(WEEK + PERIOD_QUERY_OFFSET) := 'o';
    end if;
    PERIOD_QUERY_OFFSET := PERIOD_QUERY_OFFSET + 1;

  end loop;
end;

-- Assert the query

ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
ASSERT
(KB        => PERIOD_QUERY,
 IN_CLAUSE => PERIOD_QUERY_STRING);

-- Ask the query

ESE_PERIOD_INFERENCE_PACKAGE.
SOLVE.
START
(THIS_QUERY => PERIOD_QUERY,
 THIS_KB    => PERIOD_KB);

loop

  -- Get the result

  ESE_PERIOD_INFERENCE_PACKAGE.
  CONTROL.
  GET_RESULT
  (OUT_LIST => VALUE_LIST);

  -- Check for No periods

  if SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     IS_EQUAL
     (LEFT  => "No",
      RIGHT => SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               ITEM_AT
               (LIST        => VALUE_LIST,
                NODE_NUMBER => 1)) then

    raise PERIODS_NOT_AVAILABLE;

  end if;

  -- Analyse response
```

```
EXTRACT_ONE_PERIOD
(PERIOD     => PERIOD,
 DAY        => DAY,
 OLD_CLAUSE => OLD_CLAUSE_STRING,
 NEW_CLAUSE => NEW_CLAUSE_STRING,
 VALUE_LIST => VALUE_LIST);

-- Is this solution acceptable?

declare
  COMMON_QUERY_OFFSET : POSITIVE := 18;
begin

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      COMMON_QUERY_STRING(WEEK + COMMON_QUERY_OFFSET) := 'o';
    end if;
    COMMON_QUERY_OFFSET := COMMON_QUERY_OFFSET + 1;

  end loop;
end;

COMMON_QUERY_STRING(8 .. 10) := OLD_CLAUSE_STRING(8 .. 10);
COMMON_QUERY_STRING(12 .. 13) := OLD_CLAUSE_STRING(12 .. 13);

if COMMON_MODULES_AVAILABLE
   (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR        => ACTIVITY_PTR) and then
   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST              => ACTIVITY_PTR.
                               STAFF_LIST,
   MODULE                   => ACTIVITY_PTR.
                               FACT,
   DAY                      => DAY,
   PERIOD                   => PERIOD,
   WEEK_ARRAY               => WEEKS_REQUIRED,
   STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                               STAFF_FRAME_BASE_RECORD) then
  PERIOD_1 := PERIOD;
  THIS_DAY := DAY;

  -- Release solve if no more

  ESE_PERIOD_INFERENCE_PACKAGE.
  CONTROL.
  NO_MORE;

  -- Retract current clause

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  RETRACT
  (CLAUSE => OLD_CLAUSE_STRING,
   KB     => PERIOD_KB);

  -- Assert updated clause
```

```
      if NEW_CLAUSE_STRING(18 .. 40) =
         "(x,x,x,x,x,x,x,x,x,x,x)" then

        ESE_PERIOD_INFERENCE_PACKAGE.
        LOGIC_KB.
        ASSERT
        (IN_CLAUSE => NEW_CLAUSE_STRING,
         KB        => ESE_PERIOD_KB);

      else

        ESE_PERIOD_INFERENCE_PACKAGE.
        LOGIC_KB.
        ASSERT
        (IN_CLAUSE  => NEW_CLAUSE_STRING,
         KB         => ESE_PERIOD_KB,
         AT_BACK_OF => FALSE);

      end if;

      -- Do same for all common modules

      ADJUST_COMMON_MODULE_PERIODS
      (DAY             => OLD_CLAUSE_STRING(8 .. 10),
       PERIOD          => OLD_CLAUSE_STRING(12 .. 13),
       WEEKS_REQUIRED  => WEEKS_REQUIRED,
       ACTIVITY_PTR    => ACTIVITY_PTR);

       STAFF_PACKAGE.
       MAKE_ALL_STAFF_BUSY
       (STAFF_LIST              => ACTIVITY_PTR.
                                   STAFF_LIST,
       MODULE                   => ACTIVITY_PTR.
                                   FACT,
       DAY                      => DAY,
       PERIOD                   => PERIOD,
       WEEK_ARRAY               => WEEKS_REQUIRED,
       STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                                   STAFF_FRAME_BASE_RECORD);

      SELECTED := TRUE;

      exit;

    else

      ESE_PERIOD_INFERENCE_PACKAGE.
      CONTROL.
      GET_MORE;

    end if;

  end loop;
-------------------------------------------------------------------
exception
  when PERIODS_NOT_AVAILABLE =>
    SELECTED := FALSE;
    TEXT_IO.PUT_LINE
    ("Exception PERIODS_NOT_AVALABLE in PERIOD_KS_PACKAGE " &
```

```
      "at SELECT_1_ESE_PERIOD");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_KS_PACKAGE " &
     "at SELECT_1_ESE_PERIOD");
-------------------------------------------------------------------
end SELECT_1_ESE_PERIOD;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Select_2_ese_periods identifies two adjacent periods/day that
-- satisfies
-- the current requirement
--
-------------------------------------------------------------------
procedure SELECT_2_ESE_PERIODS
-------------------------------------------------------------------
(PERIOD_1        :      out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
 PERIOD_2        :      out TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
 THIS_DAY        :      out TIMETABLE_TYPES_PACKAGE.
                            DAY_TYPE;
 WEEKS_REQUIRED :   in      TIMETABLE_TYPES_PACKAGE.
                            WEEK_ARRAY_TYPE;
 PERIOD_KB       :   in out ESE_PERIOD_INFERENCE_PACKAGE.
                            LOGIC_KB.
                            KB_RECORD;
 ACTIVITY_PTR    : in       TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_NODE_PTR_TYPE;
 SELECTED        :      out BOOLEAN) is
-------------------------------------------------------------------

  PERIOD_QUERY_STRING :
  STANDARD.
  STRING(1 .. 186) :=
  "two_periods" &
  "(DAY,FIRST_PERIOD,SECOND_PERIOD," &
  "wks(P1W1 ,P1W2 ,P1W3 ,P1W4 ,P1W5 ,P1W6 ,P1W7 ,P1W8" &
  " ,P1W9 ,P1W10,P1W11)," &
  "wks(P2W1 ,P2W2 ,P2W3 ,P2W4 ,P2W5 ,P2W6 ,P2W7 ,P2W8" &
  " ,P2W9 ,P2W10,P2W11)).";

  DYNAMIC_PERIOD_QUERY_STRING : SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                STRING;

  COMMON_QUERY_STRING :
  STANDARD.
  STRING(1 .. 42) := "period(    ,   ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

  OLD_P1_STRING    : STANDARD.
                     STRING(1 .. 42) :=
                     "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

  NEW_P1_STRING    : STANDARD.
                     STRING(1 .. 42) :=
                     "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";
```

```
  OLD_P2_STRING    : STANDARD.
                     STRING(1 .. 42) :=
                     "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

  NEW_P2_STRING    : STANDARD.
                     STRING(1 .. 42) :=
                     "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

  PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;

  PERIOD1 : TIMETABLE_TYPES_PACKAGE.
            PERIOD_NUMBER_TYPE;
  PERIOD2 : TIMETABLE_TYPES_PACKAGE.
            PERIOD_NUMBER_TYPE;
  DAY     : TIMETABLE_TYPES_PACKAGE.
            DAY_TYPE;
-------------------------------------------------------------------------
  PERIODS_NOT_AVAILABLE : exception;
-------------------------------------------------------------------------
begin

  -- Set the query string where 'o' signifies week required

  declare
    PERIOD_1_QUERY_OFFSET : POSITIVE := 48;
    PERIOD_2_QUERY_OFFSET : POSITIVE := 119;
  begin

    for WEEK in TIMETABLE_TYPES_PACKAGE.
                WEEK_NUMBER_TYPE
    loop

      if WEEKS_REQUIRED(WEEK) = TRUE then

        PERIOD_QUERY_STRING
        (PERIOD_1_QUERY_OFFSET .. PERIOD_1_QUERY_OFFSET + 4 )
        := "o    ";

        PERIOD_QUERY_STRING
        (PERIOD_2_QUERY_OFFSET .. PERIOD_2_QUERY_OFFSET + 4 )
        := "o    ";
      end if;
      PERIOD_1_QUERY_OFFSET := PERIOD_1_QUERY_OFFSET + 6;
      PERIOD_2_QUERY_OFFSET := PERIOD_2_QUERY_OFFSET + 6;

    end loop;

  end;

  STRIP_SPACES
  (IN_STANDARD_STRING => PERIOD_QUERY_STRING,
   OUT_DYNAMIC_STRING => DYNAMIC_PERIOD_QUERY_STRING);
```

```
-- Assert the query

ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
ASSERT
(KB        => PERIOD_QUERY,
 IN_CLAUSE => SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.
              SUBSTRING_OF
              (THE_STRING => DYNAMIC_PERIOD_QUERY_STRING));

-- Ask the query

ESE_PERIOD_INFERENCE_PACKAGE.
SOLVE.
START
(THIS_QUERY => PERIOD_QUERY,
 THIS_KB    => PERIOD_KB);

loop

  -- Get the result

  ESE_PERIOD_INFERENCE_PACKAGE.
  CONTROL.
  GET_RESULT
  (OUT_LIST => VALUE_LIST);

  -- Check for No periods

  if SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     IS_EQUAL
     (LEFT  => "No",
      RIGHT => SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               ITEM_AT
               (LIST        => VALUE_LIST,
                NODE_NUMBER => 1)) then

    raise PERIODS_NOT_AVAILABLE;

  end if;

  -- Analyse response

  EXTRACT_TWO_PERIODS
  (PERIOD_1       => PERIOD1,
   PERIOD_2       => PERIOD2,
   THIS_DAY       => DAY,
   OLD_P1_CLAUSE  => OLD_P1_STRING,
   OLD_P2_CLAUSE  => OLD_P2_STRING,
   NEW_P1_CLAUSE  => NEW_P1_STRING,
   NEW_P2_CLAUSE  => NEW_P2_STRING,
   VALUE_LIST     => VALUE_LIST);

  -- Is this solution acceptable?

  declare
    COMMON_QUERY_OFFSET : POSITIVE := 18;
```

```
    begin

      for WEEK in TIMETABLE_TYPES_PACKAGE.
                  WEEK_NUMBER_TYPE
      loop

        if WEEKS_REQUIRED(WEEK) = TRUE then
          COMMON_QUERY_STRING(WEEK + COMMON_QUERY_OFFSET) := 'o';
        end if;
        COMMON_QUERY_OFFSET := COMMON_QUERY_OFFSET + 1;

      end loop;
    end;

    COMMON_QUERY_STRING(8 .. 10) := OLD_P1_STRING(8 .. 10);
    COMMON_QUERY_STRING(12 .. 13) := OLD_P1_STRING(12 .. 13);

    if COMMON_MODULES_AVAILABLE
       (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
        ACTIVITY_PTR        => ACTIVITY_PTR)  and then
       STAFF_PACKAGE.
       ALL_STAFF_FREE
       (STAFF_LIST              => ACTIVITY_PTR.
                                   STAFF_LIST,
       MODULE                   => ACTIVITY_PTR.
                                   FACT,
       DAY                      => DAY,
       PERIOD                   => PERIOD1,
       WEEK_ARRAY               => WEEKS_REQUIRED,
       STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                   STAFF_FRAME_BASE_RECORD) then

      COMMON_QUERY_STRING(12 .. 13) := OLD_P2_STRING(12 .. 13);

      if COMMON_MODULES_AVAILABLE
          (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
           ACTIVITY_PTR        => ACTIVITY_PTR)  and then
         STAFF_PACKAGE.
         ALL_STAFF_FREE
         (STAFF_LIST              => ACTIVITY_PTR.
                                     STAFF_LIST,
         MODULE                   => ACTIVITY_PTR.
                                     FACT,
         DAY                      => DAY,
         PERIOD                   => PERIOD2,
         WEEK_ARRAY               => WEEKS_REQUIRED,
         STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                     STAFF_FRAME_BASE_RECORD) then

         PERIOD_1 := PERIOD1;
         PERIOD_2 := PERIOD2;
         THIS_DAY := DAY;

        -- Release solve if no more

        ESE_PERIOD_INFERENCE_PACKAGE.
        CONTROL.
        NO_MORE;

        -- Retract current clauses
```

```
ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_P1_STRING,
 KB     => PERIOD_KB);

ESE_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_P2_STRING,
 KB     => PERIOD_KB);

-- Assert updated clause

if NEW_P1_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P1_STRING,
   KB        => ESE_PERIOD_KB);

else

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE  => NEW_P1_STRING,
   KB         => ESE_PERIOD_KB,
   AT_BACK_OF => FALSE);

end if;

if NEW_P2_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P2_STRING,
   KB        => ESE_PERIOD_KB);

else

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE  => NEW_P2_STRING,
   KB         => ESE_PERIOD_KB,
   AT_BACK_OF => FALSE);

end if;

-- Do same for all common modules

ADJUST_COMMON_MODULE_PERIODS
(DAY            => OLD_P1_STRING(8 .. 10),
 PERIOD         => OLD_P1_STRING(12 .. 13),
```

```
            WEEKS_REQUIRED => WEEKS_REQUIRED,
            ACTIVITY_PTR   => ACTIVITY_PTR);

          STAFF_PACKAGE.
          MAKE_ALL_STAFF_BUSY
          (STAFF_LIST              => ACTIVITY_PTR.
                                      STAFF_LIST,
          MODULE                   => ACTIVITY_PTR.
                                      FACT,
          DAY                      => DAY,
          PERIOD                   => PERIOD1,
          WEEK_ARRAY               => WEEKS_REQUIRED,
          STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                      STAFF_FRAME_BASE_RECORD);

          ADJUST_COMMON_MODULE_PERIODS
          (DAY             => OLD_P2_STRING(8 .. 10),
           PERIOD          => OLD_P2_STRING(12 .. 13),
           WEEKS_REQUIRED => WEEKS_REQUIRED,
           ACTIVITY_PTR    => ACTIVITY_PTR);

          STAFF_PACKAGE.
          MAKE_ALL_STAFF_BUSY
          (STAFF_LIST              => ACTIVITY_PTR.
                                      STAFF_LIST,
          MODULE                   => ACTIVITY_PTR.
                                      FACT,
          DAY                      => DAY,
          PERIOD                   => PERIOD2,
          WEEK_ARRAY               => WEEKS_REQUIRED,
          STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                      STAFF_FRAME_BASE_RECORD);

          SELECTED := TRUE;

          exit;
        end if;

      else

        ESE_PERIOD_INFERENCE_PACKAGE.
        CONTROL.
        GET_MORE;

      end if;
    end loop;
  ------------------------------------------------------------------
  exception
    when PERIODS_NOT_AVAILABLE =>
      SELECTED := FALSE;
      TEXT_IO.PUT_LINE
      ("Exception PERIODS_NOT_AVALABLE in PERIOD_KS_PACKAGE " &
       "at SELECT_2_ESE_PERIODS");
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_KS_PACKAGE " &
       "at SELECT_2_ESE_PERIODS");
  ------------------------------------------------------------------
  end SELECT_2_ESE_PERIODS;
  ------------------------------------------------------------------
```

```
------------------------------------------------------------------
--
-- Select_3_ese_periods identifies two adjacent periods/day that
-- satisfies the current requirement
--
------------------------------------------------------------------
procedure SELECT_3_ESE_PERIODS
------------------------------------------------------------------
(PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 PERIOD_3        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                         DAY_TYPE;
 WEEKS_REQUIRED :  in      TIMETABLE_TYPES_PACKAGE.
                         WEEK_ARRAY_TYPE;
 PERIOD_KB       :  in out ESE_PERIOD_INFERENCE_PACKAGE.
                         LOGIC_KB.
                         KB_RECORD;
 ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
 SELECTED        :     out BOOLEAN) is
------------------------------------------------------------------

  PERIOD_QUERY_STRING :
  STANDARD.
  STRING(1 .. 272) :=
  "three_periods" &
  "(DAY,FIRST_PERIOD,SECOND_PERIOD,THIRD_PERIOD," &
  "wks(P1W1 ,P1W2 ,P1W3 ,P1W4 ,P1W5 ,P1W6 ,P1W7 ,P1W8" &
  " ,P1W9 ,P1W10,P1W11)," &
  "wks(P2W1 ,P2W2 ,P2W3 ,P2W4 ,P2W5 ,P2W6 ,P2W7 ,P2W8" &
  " ,P2W9 ,P2W10,P2W11)," &
  "wks(P3W1 ,P3W2 ,P3W3 ,P3W4 ,P3W5 ,P3W6 ,P3W7 ,P3W8" &
  " ,P3W9 ,P3W10,P3W11)).";

  DYNAMIC_PERIOD_QUERY_STRING : SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
                                STRING;

  COMMON_QUERY_STRING :
  STANDARD.
  STRING(1 .. 42) :=
  "period(   ,   ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

  OLD_P1_STRING   : STANDARD.
                    STRING(1 .. 42) :=
                    "period(   ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

  NEW_P1_STRING   : STANDARD.
                    STRING(1 .. 42) :=
                    "period(   ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

  OLD_P2_STRING   : STANDARD.
                    STRING(1 .. 42) :=
                    "period(   ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";
```

```
   NEW_P2_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x)).";

   OLD_P3_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o)).";

   NEW_P3_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x)).";

   PERIOD_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  KB_RECORD;

   VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_LIST_PACKAGE.
                LIST_TYPE;

   PERIOD1 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   PERIOD2 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   PERIOD3 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;

   DAY     : TIMETABLE_TYPES_PACKAGE.
             DAY_TYPE;

   PERIODS_NOT_AVAILABLE : exception;

-------------------------------------------------------------------
begin

   -- Set the query string where 'o' signifies week required

   declare
     PERIOD_1_QUERY_OFFSET : POSITIVE := 63;
     PERIOD_2_QUERY_OFFSET : POSITIVE := 134;
     PERIOD_3_QUERY_OFFSET : POSITIVE := 205;
   begin

     for WEEK in TIMETABLE_TYPES_PACKAGE.
                 WEEK_NUMBER_TYPE
     loop

       if WEEKS_REQUIRED(WEEK) = TRUE then

         PERIOD_QUERY_STRING
         (PERIOD_1_QUERY_OFFSET .. PERIOD_1_QUERY_OFFSET + 4 )
         := "o    ";

         PERIOD_QUERY_STRING
         (PERIOD_2_QUERY_OFFSET .. PERIOD_2_QUERY_OFFSET + 4 )
         := "o    ";

         PERIOD_QUERY_STRING
         (PERIOD_3_QUERY_OFFSET .. PERIOD_3_QUERY_OFFSET + 4 )
```

```
        := "o    ";

      end if;
      PERIOD_1_QUERY_OFFSET := PERIOD_1_QUERY_OFFSET + 6;
      PERIOD_2_QUERY_OFFSET := PERIOD_2_QUERY_OFFSET + 6;
      PERIOD_3_QUERY_OFFSET := PERIOD_3_QUERY_OFFSET + 6;

    end loop;

  end;

  STRIP_SPACES
  (IN_STANDARD_STRING => PERIOD_QUERY_STRING,
   OUT_DYNAMIC_STRING => DYNAMIC_PERIOD_QUERY_STRING);

  -- Assert the query

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (KB        => PERIOD_QUERY,
   IN_CLAUSE => SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                SUBSTRING_OF
                (THE_STRING => DYNAMIC_PERIOD_QUERY_STRING));

  -- Ask the query

  ESE_PERIOD_INFERENCE_PACKAGE.
  SOLVE.
  START
  (THIS_QUERY => PERIOD_QUERY,
   THIS_KB    => PERIOD_KB);

  loop

    -- Get the result

    ESE_PERIOD_INFERENCE_PACKAGE.
    CONTROL.
    GET_RESULT
    (OUT_LIST => VALUE_LIST);

    -- Check for No periods

    if SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       IS_EQUAL
       (LEFT  => "No",
        RIGHT => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING_LIST_PACKAGE.
                 ITEM_AT
                 (LIST        => VALUE_LIST,
                  NODE_NUMBER => 1)) then

      raise PERIODS_NOT_AVAILABLE;

    end if;

    -- Analyse response
```

```
EXTRACT_THREE_PERIODS
(PERIOD_1        => PERIOD1,
 PERIOD_2        => PERIOD2,
 PERIOD_3        => PERIOD3,
 THIS_DAY        => DAY,
 OLD_P1_CLAUSE   => OLD_P1_STRING,
 OLD_P2_CLAUSE   => OLD_P2_STRING,
 OLD_P3_CLAUSE   => OLD_P3_STRING,
 NEW_P1_CLAUSE   => NEW_P1_STRING,
 NEW_P2_CLAUSE   => NEW_P2_STRING,
 NEW_P3_CLAUSE   => NEW_P3_STRING,
 VALUE_LIST      => VALUE_LIST);

-- Is this solution acceptable?

declare
  COMMON_QUERY_OFFSET : POSITIVE := 18;
begin

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      COMMON_QUERY_STRING(WEEK + COMMON_QUERY_OFFSET) := 'o';
    end if;
    COMMON_QUERY_OFFSET := COMMON_QUERY_OFFSET + 1;

  end loop;
end;

COMMON_QUERY_STRING(8 .. 10) := OLD_P1_STRING(8 .. 10);
COMMON_QUERY_STRING(12 .. 13) := OLD_P1_STRING(12 ..13);

if COMMON_MODULES_AVAILABLE
   (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR        => ACTIVITY_PTR) and then
   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST              => ACTIVITY_PTR..
                               STAFF_LIST,
   MODULE                   => ACTIVITY_PTR.
                               FACT,
   DAY                      => DAY,
   PERIOD                   => PERIOD1,
   WEEK_ARRAY               => WEEKS_REQUIRED,
   STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                               STAFF_FRAME_BASE_RECORD) then

  COMMON_QUERY_STRING(12 .. 13) := OLD_P2_STRING(12 .. 13);

  if COMMON_MODULES_AVAILABLE
   (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR        => ACTIVITY_PTR) and then

   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST              => ACTIVITY_PTR.
                               STAFF_LIST,
```

```
MODULE                    => ACTIVITY_PTR.
                             FACT,
DAY                       => DAY,
PERIOD                    => PERIOD2,
WEEK_ARRAY                => WEEKS_REQUIRED,
STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                             STAFF_FRAME_BASE_RECORD) then

 COMMON_QUERY_STRING(12 .. 13) := OLD_P3_STRING(12 ..13);

 if COMMON_MODULES_AVAILABLE
  (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
   ACTIVITY_PTR        => ACTIVITY_PTR) and then

 STAFF_PACKAGE.
 ALL_STAFF_FREE
 (STAFF_LIST              => ACTIVITY_PTR.
                             STAFF_LIST,
 MODULE                   => ACTIVITY_PTR.
                             FACT,
 DAY                      => DAY,
 PERIOD                   => PERIOD3,
 WEEK_ARRAY               => WEEKS_REQUIRED,
 STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                             STAFF_FRAME_BASE_RECORD) then

   PERIOD_1 := PERIOD1;
   PERIOD_2 := PERIOD2;
   PERIOD_3 := PERIOD3;
   THIS_DAY := DAY;

   -- Release solve if no more

   ESE_PERIOD_INFERENCE_PACKAGE.
   CONTROL.
   NO_MORE;

   -- Retract current clauseS

   ESE_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   RETRACT
   (CLAUSE => OLD_P1_STRING,
    KB     => PERIOD_KB);

   ESE_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   RETRACT
   (CLAUSE => OLD_P2_STRING,
    KB     => PERIOD_KB);

   ESE_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   RETRACT
   (CLAUSE => OLD_P3_STRING,
    KB     => PERIOD_KB);


   -- Assert updated clause
```

```
if NEW_P1_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P1_STRING,
   KB        => ESE_PERIOD_KB);

else

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE  => NEW_P1_STRING,
   KB         => ESE_PERIOD_KB,
   AT_BACK_OF => FALSE);

end if;

if NEW_P2_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P2_STRING,
   KB        => ESE_PERIOD_KB);

else

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE  => NEW_P2_STRING,
   KB         => ESE_PERIOD_KB,
   AT_BACK_OF => FALSE);

end if;

if NEW_P3_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P3_STRING,
   KB        => ESE_PERIOD_KB);

else

  ESE_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE  => NEW_P3_STRING,
   KB         => ESE_PERIOD_KB,
   AT_BACK_OF => FALSE);

end if;
```

```
-- Do same for all common modules

ADJUST_COMMON_MODULE_PERIODS
(DAY             => OLD_P1_STRING(8 .. 10),
 PERIOD          => OLD_P1_STRING(12 .. 13),
 WEEKS_REQUIRED => WEEKS_REQUIRED,
 ACTIVITY_PTR    => ACTIVITY_PTR);

STAFF_PACKAGE.
MAKE_ALL_STAFF_BUSY
(STAFF_LIST              => ACTIVITY_PTR.
                            STAFF_LIST,
MODULE                   => ACTIVITY_PTR.
                            FACT,
DAY                      => DAY,
PERIOD                   => PERIOD1,
WEEK_ARRAY               => WEEKS_REQUIRED,
STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                            STAFF_FRAME_BASE_RECORD);

ADJUST_COMMON_MODULE_PERIODS
(DAY             => OLD_P2_STRING(8 .. 10),
 PERIOD          => OLD_P2_STRING(12 .. 13),
 WEEKS_REQUIRED => WEEKS_REQUIRED,
 ACTIVITY_PTR    => ACTIVITY_PTR);

STAFF_PACKAGE.
MAKE_ALL_STAFF_BUSY
(STAFF_LIST              => ACTIVITY_PTR.
                            STAFF_LIST,
MODULE                   => ACTIVITY_PTR.
                            FACT,
DAY                      => DAY,
PERIOD                   => PERIOD2,
WEEK_ARRAY               => WEEKS_REQUIRED,
STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                            STAFF_FRAME_BASE_RECORD);

ADJUST_COMMON_MODULE_PERIODS
(DAY             => OLD_P3_STRING(8 .. 10),
 PERIOD          => OLD_P3_STRING(12 .. 13),
 WEEKS_REQUIRED => WEEKS_REQUIRED,
 ACTIVITY_PTR    => ACTIVITY_PTR);

STAFF_PACKAGE.
MAKE_ALL_STAFF_BUSY
(STAFF_LIST              => ACTIVITY_PTR.
                            STAFF_LIST,
MODULE                   => ACTIVITY_PTR.
                            FACT,
DAY                      => DAY,
PERIOD                   => PERIOD3,
WEEK_ARRAY               => WEEKS_REQUIRED,
STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                            STAFF_FRAME_BASE_RECORD);

SELECTED := TRUE;

exit;
```

```
          end if;
        end if;

      else

        ESE_PERIOD_INFERENCE_PACKAGE.
        CONTROL.
        GET_MORE;

      end if;
    end loop;
  ------------------------------------------------------------------------
  exception
    when PERIODS_NOT_AVAILABLE =>
      SELECTED := FALSE;
      TEXT_IO.PUT_LINE
      ("Exception PERIODS_NOT_AVALABLE in PERIOD_KS_PACKAGE " &
       "at SELECT_3_ESE_PERIODS");
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_KS_PACKAGE " &
       "at SELECT_3_ESE_PERIODS");
  ------------------------------------------------------------------------
  end SELECT_3_ESE_PERIODS;
  ------------------------------------------------------------------------

------------------------------------------------------------------------
end PERIOD_ESE_PACKAGE;
------------------------------------------------------------------------
```

Period IT

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_IT_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides the blackboard transformation
--             from the degree activities level to the day level by
--             allocating periods to the degree activities
--
-------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE;
use  PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE;
-------------------------------------------------------------------
package PERIOD_IT_PACKAGE is
-------------------------------------------------------------------

  -----------------------------------------------------------------
  procedure SELECT_1_IT_PERIOD
  -----------------------------------------------------------------
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
   PERIOD_KB       :  in out IT_PERIOD_INFERENCE_PACKAGE.
                           LOGIC_KB.
                           KB_RECORD;
   ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
   SELECTED        :     out BOOLEAN);
  -----------------------------------------------------------------

  -----------------------------------------------------------------
  procedure SELECT_2_IT_PERIODS
  -----------------------------------------------------------------
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
```

```
                                WEEK_ARRAY_TYPE;
   PERIOD_KB        :  in out IT_PERIOD_INFERENCE_PACKAGE.
                              LOGIC_KB.
                              KB_RECORD;
   ACTIVITY_PTR     : in      TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_NODE_PTR_TYPE;
   SELECTED         :     out BOOLEAN);
  ------------------------------------------------------------------

  ------------------------------------------------------------------
  procedure SELECT_3_IT_PERIODS
  ------------------------------------------------------------------
  (PERIOD_1         :     out TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   PERIOD_2         :     out TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   PERIOD_3         :     out TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   THIS_DAY         :     out TIMETABLE_TYPES_PACKAGE.
                              DAY_TYPE;
   WEEKS_REQUIRED :  in      TIMETABLE_TYPES_PACKAGE.
                              WEEK_ARRAY_TYPE;
   PERIOD_KB        :  in out IT_PERIOD_INFERENCE_PACKAGE.
                              LOGIC_KB.
                              KB_RECORD;
   ACTIVITY_PTR     : in      TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_NODE_PTR_TYPE;
   SELECTED         :     out BOOLEAN);
  ------------------------------------------------------------------

------------------------------------------------------------------
end PERIOD_IT_PACKAGE;
------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_IT_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             CRanfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides the blackboard transformation
--             from the
--             degree activity level to the day level by allocating
--             periods to the degree activities
--
-------------------------------------------------------------------
package body PERIOD_IT_PACKAGE is
-------------------------------------------------------------------
  use TEXT_IO;
  -----------------------------------------------------------------
  --
  -- Select period identifies a period/day that satisfies the
  -- current
  -- requirement
  --
  -----------------------------------------------------------------
  procedure SELECT_1_IT_PERIOD
  -----------------------------------------------------------------
  (PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE;
   THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                           DAY_TYPE;
   WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                           WEEK_ARRAY_TYPE;
   PERIOD_KB       :  in out IT_PERIOD_INFERENCE_PACKAGE.
                           LOGIC_KB.
                           KB_RECORD;
   ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                           TIMETABLE_NODE_PTR_TYPE;
   SELECTED        :     out BOOLEAN) is
  -----------------------------------------------------------------

    PERIOD_QUERY_STRING :
    STANDARD.
    STRING
    (1 .. 43) := "one_period(A,B,wks(C,D,E,F,G,H,I,J,K,L,M)).";

    COMMON_QUERY_STRING :
    STANDARD.
    STRING(1 .. 42) :=
    "period(   ,   ,wks(_,_,_,_,_,_,_,_,_,_,_,_,_)).";

    OLD_CLAUSE_STRING   :
    STANDARD.
    STRING(1 .. 42) := "period(   ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o,o)).";

    NEW_CLAUSE_STRING   :
    STANDARD.
    STRING(1 .. 42) := "period(   ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x,x)).";

    PERIOD_QUERY : IT_PERIOD_INFERENCE_PACKAGE.
                   LOGIC_KB.
                   KB_RECORD;
```

```
  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;

  PERIOD1 : TIMETABLE_TYPES_PACKAGE.
            PERIOD_NUMBER_TYPE;
  DAY     : TIMETABLE_TYPES_PACKAGE.
            DAY_TYPE;

  PERIODS_NOT_AVAILABLE : exception;

-------------------------------------------------------------------
begin

  -- Set the query string where 'o' signifies week required

  declare
    PERIOD_QUERY_OFFSET : POSITIVE := 19;
  begin

    for WEEK in TIMETABLE_TYPES_PACKAGE.
                WEEK_NUMBER_TYPE
    loop

      if WEEKS_REQUIRED(WEEK) = TRUE then
        PERIOD_QUERY_STRING(WEEK + PERIOD_QUERY_OFFSET) := 'o';
      end if;
      PERIOD_QUERY_OFFSET := PERIOD_QUERY_OFFSET + 1;

    end loop;
  end;

  -- Assert the query

  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (KB         => PERIOD_QUERY,
   IN_CLAUSE => PERIOD_QUERY_STRING);

  -- Ask the query

  IT_PERIOD_INFERENCE_PACKAGE.
  SOLVE.
  START
  (THIS_QUERY => PERIOD_QUERY,
   THIS_KB    => PERIOD_KB);

  loop

    -- Get the result

    IT_PERIOD_INFERENCE_PACKAGE.
    CONTROL.
    GET_RESULT
    (OUT_LIST => VALUE_LIST);

    -- Check for No periods

    if SYSTEM_TYPES_PACKAGE.
```

```
    DYNAMIC_STRING.
    IS_EQUAL
    (LEFT  => "No",
     RIGHT => SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING_LIST_PACKAGE.
              ITEM_AT
              (LIST        => VALUE_LIST,
               NODE_NUMBER => 1)) then

  raise PERIODS_NOT_AVAILABLE;

end if;

-- Analyse response

EXTRACT_ONE_PERIOD
(PERIOD     => PERIOD1,
 DAY        => DAY,
 OLD_CLAUSE => OLD_CLAUSE_STRING,
 NEW_CLAUSE => NEW_CLAUSE_STRING,
 VALUE_LIST => VALUE_LIST);

-- Is this solution acceptable?

declare
  COMMON_QUERY_OFFSET : POSITIVE := 18;
begin

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      COMMON_QUERY_STRING(WEEK + COMMON_QUERY_OFFSET) := 'o';
    end if;
    COMMON_QUERY_OFFSET := COMMON_QUERY_OFFSET + 1;

  end loop;
end;

COMMON_QUERY_STRING(8 .. 10) := OLD_CLAUSE_STRING(8 .. 10);
COMMON_QUERY_STRING(12 .. 13) := OLD_CLAUSE_STRING(12 ..13);

if COMMON_MODULES_AVAILABLE
   (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR        => ACTIVITY_PTR) and then
   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST              => ACTIVITY_PTR.
                               STAFF_LIST,
   MODULE                   => ACTIVITY_PTR.
                               FACT,
   DAY                      => DAY,
   PERIOD                   => PERIOD1,
   WEEK_ARRAY               => WEEKS_REQUIRED,
   STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                               STAFF_FRAME_BASE_RECORD) then

  PERIOD_1 := PERIOD1;
  THIS_DAY := DAY;
```

```
-- Release solve if no more

IT_PERIOD_INFERENCE_PACKAGE.
CONTROL.
NO_MORE;

-- Retract current clause

IT_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_CLAUSE_STRING,
 KB     => PERIOD_KB);

-- Assert updated clause

  if NEW_CLAUSE_STRING(18 .. 40) =
     "(x,x,x,x,x,x,x,x,x,x,x)" then

    IT_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE => NEW_CLAUSE_STRING,
     KB        => IT_PERIOD_KB);

  else

    IT_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    ASSERT
    (IN_CLAUSE  => NEW_CLAUSE_STRING,
     KB         => IT_PERIOD_KB,
     AT_BACK_OF => FALSE);

  end if;

-- Do same for all common modules

ADJUST_COMMON_MODULE_PERIODS
(DAY             => OLD_CLAUSE_STRING(8 .. 10),
 PERIOD          => OLD_CLAUSE_STRING(12 .. 13),
 WEEKS_REQUIRED  => WEEKS_REQUIRED,
 ACTIVITY_PTR    => ACTIVITY_PTR);

 STAFF_PACKAGE.
 MAKE_ALL_STAFF_BUSY
 (STAFF_LIST              => ACTIVITY_PTR.
                             STAFF_LIST,
 MODULE                   => ACTIVITY_PTR.
                             FACT,
 DAY                      => DAY,
 PERIOD                   => PERIOD1,
 WEEK_ARRAY               => WEEKS_REQUIRED,
 STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                             STAFF_FRAME_BASE_RECORD);

SELECTED := TRUE;

exit;
```

```
    else

      IT_PERIOD_INFERENCE_PACKAGE.
      CONTROL.
      GET_MORE;

    end if;

  end loop;
-------------------------------------------------------------------
exception
  when PERIODS_NOT_AVAILABLE =>
    SELECTED := FALSE;
    TEXT_IO.PUT_LINE
    ("Exception PERIODS_NOT_AVALABLE in PERIOD_KS_PACKAGE " &
     "at SELECT_1_IT_PERIOD");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_KS_PACKAGE " &
     "at SELECT_1_IT_PERIOD");
-------------------------------------------------------------------
end SELECT_1_IT_PERIOD;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Select_2_it_periods identifies two adjacent periods/day that
-- satisfies the current requirement
--
-------------------------------------------------------------------
procedure SELECT_2_IT_PERIODS
-------------------------------------------------------------------
(PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                         DAY_TYPE;
 WEEKS_REQUIRED :  in      TIMETABLE_TYPES_PACKAGE.
                         WEEK_ARRAY_TYPE;
 PERIOD_KB       :  in out IT_PERIOD_INFERENCE_PACKAGE.
                         LOGIC_KB.
                         KB_RECORD;
 ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
 SELECTED        :     out BOOLEAN) is
-------------------------------------------------------------------

  PERIOD_QUERY_STRING :
  STANDARD.
  STRING(1 .. 186) :=
  "two_periods" &
  "(DAY,FIRST_PERIOD,SECOND_PERIOD," &
  "wks(P1W1 ,P1W2 ,P1W3 ,P1W4 ,P1W5 ,P1W6 ,P1W7 ,P1W8" &
  " ,P1W9 ,P1W10,P1W11)," &
  "wks(P2W1 ,P2W2 ,P2W3 ,P2W4 ,P2W5 ,P2W6 ,P2W7 ,P2W8" &
  " ,P2W9 ,P2W10,P2W11)).";
```

```
   DYNAMIC_PERIOD_QUERY_STRING : SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING.
                                 STRING;

   COMMON_QUERY_STRING :
   STANDARD.
   STRING(1 .. 42) := "period(    ,   ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

   OLD_P1_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

   NEW_P1_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

   OLD_P2_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

   NEW_P2_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

   PERIOD_QUERY : IT_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  KB_RECORD;

   VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_LIST_PACKAGE.
                LIST_TYPE;

   PERIOD1 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   PERIOD2 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   DAY     : TIMETABLE_TYPES_PACKAGE.
             DAY_TYPE;

   PERIODS_NOT_AVAILABLE : exception;

-------------------------------------------------------------------
begin

   -- Set the query string where 'o' signifies week required

   declare
     PERIOD_1_QUERY_OFFSET : POSITIVE := 48;
     PERIOD_2_QUERY_OFFSET : POSITIVE := 119;
   begin

     for WEEK in TIMETABLE_TYPES_PACKAGE.
                 WEEK_NUMBER_TYPE
     loop

       if WEEKS_REQUIRED(WEEK) = TRUE then

         PERIOD_QUERY_STRING
         (PERIOD_1_QUERY_OFFSET .. PERIOD_1_QUERY_OFFSET + 4 )
         := "o    ";
```

```
      PERIOD_QUERY_STRING
      (PERIOD_2_QUERY_OFFSET .. PERIOD_2_QUERY_OFFSET + 4 )
      := "o    ";
    end if;
    PERIOD_1_QUERY_OFFSET := PERIOD_1_QUERY_OFFSET + 6;
    PERIOD_2_QUERY_OFFSET := PERIOD_2_QUERY_OFFSET + 6;

  end loop;
end;

STRIP_SPACES
(IN_STANDARD_STRING => PERIOD_QUERY_STRING,
 OUT_DYNAMIC_STRING => DYNAMIC_PERIOD_QUERY_STRING);

-- Assert the query

IT_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
ASSERT
(KB        => PERIOD_QUERY,
 IN_CLAUSE => SYSTEM_TYPES_PACKAGE.
              DYNAMIC_STRING.
              SUBSTRING_OF
              (THE_STRING => DYNAMIC_PERIOD_QUERY_STRING));

-- Ask the query

IT_PERIOD_INFERENCE_PACKAGE.
SOLVE.
START
(THIS_QUERY => PERIOD_QUERY,
 THIS_KB    => PERIOD_KB);

loop

  -- Get the result

  IT_PERIOD_INFERENCE_PACKAGE.
  CONTROL.
  GET_RESULT
  (OUT_LIST => VALUE_LIST);

  -- Check for No periods

  if SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING.
     IS_EQUAL
     (LEFT  => "No",
      RIGHT => SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               ITEM_AT
               (LIST        => VALUE_LIST,
                NODE_NUMBER => 1)) then

    raise PERIODS_NOT_AVAILABLE;

  end if;

  -- Analyse response
```

```
EXTRACT_TWO_PERIODS
(PERIOD_1        => PERIOD1,
 PERIOD_2        => PERIOD2,
 THIS_DAY        => DAY,
 OLD_P1_CLAUSE   => OLD_P1_STRING,
 OLD_P2_CLAUSE   => OLD_P2_STRING,
 NEW_P1_CLAUSE   => NEW_P1_STRING,
 NEW_P2_CLAUSE   => NEW_P2_STRING,
 VALUE_LIST      => VALUE_LIST);

-- Is this solution acceptable?

declare
  COMMON_QUERY_OFFSET : POSITIVE := 18;
begin

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      COMMON_QUERY_STRING(WEEK + COMMON_QUERY_OFFSET) := 'o';
    end if;
    COMMON_QUERY_OFFSET := COMMON_QUERY_OFFSET + 1;

  end loop;
end;

COMMON_QUERY_STRING(8 .. 10) := OLD_P1_STRING(8 .. 10);
COMMON_QUERY_STRING(12 .. 13) := OLD_P1_STRING(12 .. 13);

if COMMON_MODULES_AVAILABLE
   (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR        => ACTIVITY_PTR) and then
   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST              => ACTIVITY_PTR.
                               STAFF_LIST,
   MODULE                   => ACTIVITY_PTR.
                               FACT,
   DAY                      => DAY,
   PERIOD                   => PERIOD1,
   WEEK_ARRAY               => WEEKS_REQUIRED,
   STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                               STAFF_FRAME_BASE_RECORD) then

  COMMON_QUERY_STRING(12 .. 13) := OLD_P2_STRING(12 .. 13);

  if COMMON_MODULES_AVAILABLE
      (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
       ACTIVITY_PTR        => ACTIVITY_PTR) and then
     STAFF_PACKAGE.
     ALL_STAFF_FREE
     (STAFF_LIST              => ACTIVITY_PTR.
                                 STAFF_LIST,
     MODULE                   => ACTIVITY_PTR.
                                 FACT,
     DAY                      => DAY,
     PERIOD                   => PERIOD2,
```

```
 WEEK_ARRAY                => WEEKS_REQUIRED,
 STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                            STAFF_FRAME_BASE_RECORD) then

PERIOD_1 := PERIOD1;
PERIOD_2 := PERIOD2;
THIS_DAY := DAY;

-- Release solve if no more

IT_PERIOD_INFERENCE_PACKAGE.
CONTROL.
NO_MORE;

-- Retract current clauseS

IT_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_P1_STRING,
 KB     => PERIOD_KB);

IT_PERIOD_INFERENCE_PACKAGE.
LOGIC_KB.
RETRACT
(CLAUSE => OLD_P2_STRING,
 KB     => PERIOD_KB);

-- Assert updated clause

if NEW_P1_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P1_STRING,
   KB        => IT_PERIOD_KB);

else

  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE  => NEW_P1_STRING,
   KB         => IT_PERIOD_KB,
   AT_BACK_OF => FALSE);

end if;

if NEW_P2_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE => NEW_P2_STRING,
   KB        => IT_PERIOD_KB);

else
```

```
        IT_PERIOD_INFERENCE_PACKAGE.
        LOGIC_KB.
        ASSERT
        (IN_CLAUSE  => NEW_P2_STRING,
         KB         => IT_PERIOD_KB,
         AT_BACK_OF => FALSE);

      end if;

      -- Do same for all common modules

      ADJUST_COMMON_MODULE_PERIODS
      (DAY             => OLD_P1_STRING(8 .. 10),
       PERIOD          => OLD_P1_STRING(12 .. 13),
       WEEKS_REQUIRED  => WEEKS_REQUIRED,
       ACTIVITY_PTR    => ACTIVITY_PTR);

      STAFF_PACKAGE.
      MAKE_ALL_STAFF_BUSY
      (STAFF_LIST              => ACTIVITY_PTR.
                                  STAFF_LIST,
      MODULE                   => ACTIVITY_PTR.
                                  FACT,
      DAY                      => DAY,
      PERIOD                   => PERIOD1,
      WEEK_ARRAY               => WEEKS_REQUIRED,
      STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                                  STAFF_FRAME_BASE_RECORD);

      ADJUST_COMMON_MODULE_PERIODS
      (DAY             => OLD_P2_STRING(8 .. 10),
       PERIOD          => OLD_P2_STRING(12 .. 13),
       WEEKS_REQUIRED  => WEEKS_REQUIRED,
       ACTIVITY_PTR    => ACTIVITY_PTR);

      STAFF_PACKAGE.
      MAKE_ALL_STAFF_BUSY
      (STAFF_LIST              => ACTIVITY_PTR.
                                  STAFF_LIST,
      MODULE                   => ACTIVITY_PTR.
                                  FACT,
      DAY                      => DAY,
      PERIOD                   => PERIOD2,
      WEEK_ARRAY               => WEEKS_REQUIRED,
      STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                                  STAFF_FRAME_BASE_RECORD);

      SELECTED := TRUE;

      exit;

    end if;

  else

    IT_PERIOD_INFERENCE_PACKAGE.
    CONTROL.
    GET_MORE;
```

```
    end if;

  end loop;

exception
  when PERIODS_NOT_AVAILABLE =>
    SELECTED := FALSE;
    TEXT_IO.PUT_LINE
    ("Exception PERIODS_NOT_AVALABLE in PERIOD_KS_PACKAGE " &
     "at SELECT_2_IT_PERIODS");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in PERIOD_KS_PACKAGE " &
     "at SELECT_2_IT_PERIODS");

-------------------------------------------------------------------
end SELECT_2_IT_PERIODS;
-------------------------------------------------------------------


-------------------------------------------------------------------
--
-- Select_3_it_periods identifies two adjacent periods/day that
-- satisfies the current requirement
--
-------------------------------------------------------------------
procedure SELECT_3_IT_PERIODS
-------------------------------------------------------------------
(PERIOD_1        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 PERIOD_2        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 PERIOD_3        :     out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
 THIS_DAY        :     out TIMETABLE_TYPES_PACKAGE.
                         DAY_TYPE;
 WEEKS_REQUIRED  :  in     TIMETABLE_TYPES_PACKAGE.
                         WEEK_ARRAY_TYPE;
 PERIOD_KB       :  in out IT_PERIOD_INFERENCE_PACKAGE.
                         LOGIC_KB.
                         KB_RECORD;
 ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
 SELECTED        :     out BOOLEAN) is
-------------------------------------------------------------------

  PERIOD_QUERY_STRING :
  STANDARD.
  STRING(1 .. 272) :=
  "three_periods" &
  "(DAY,FIRST_PERIOD,SECOND_PERIOD,THIRD_PERIOD," &
  "wks(P1W1 ,P1W2 ,P1W3 ,P1W4 ,P1W5 ,P1W6 ,P1W7 ,P1W8" &
  " ,P1W9 ,P1W10,P1W11)," &
  "wks(P2W1 ,P2W2 ,P2W3 ,P2W4 ,P2W5 ,P2W6 ,P2W7 ,P2W8" &
  " ,P2W9 ,P2W10,P2W11)," &
  "wks(P3W1 ,P3W2 ,P3W3 ,P3W4 ,P3W5 ,P3W6 ,P3W7 ,P3W8" &
  " ,P3W9 ,P3W10,P3W11)).";

  DYNAMIC_PERIOD_QUERY_STRING : SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING.
```

```
                                    STRING;

   COMMON_QUERY_STRING :
   STANDARD.
   STRING(1 .. 42) := "period(    ,   ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

   OLD_P1_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

   NEW_P1_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

   OLD_P2_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

   NEW_P2_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

   OLD_P3_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(o,o,o,o,o,o,o,o,o,o,o,o)).";

   NEW_P3_STRING    : STANDARD.
                      STRING(1 .. 42) :=
                      "period(    ,   ,wks(x,x,x,x,x,x,x,x,x,x,x,x)).";

   PERIOD_QUERY : IT_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  KB_RECORD;

   VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_LIST_PACKAGE.
                LIST_TYPE;

   PERIOD1 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   PERIOD2 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   PERIOD3 : TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE;
   DAY     : TIMETABLE_TYPES_PACKAGE.
             DAY_TYPE;

   PERIODS_NOT_AVAILABLE : exception;

-------------------------------------------------------------------
begin

   -- Set the query string where 'o' signifies week required

   declare
     PERIOD_1_QUERY_OFFSET : POSITIVE := 63;
     PERIOD_2_QUERY_OFFSET : POSITIVE := 134;
     PERIOD_3_QUERY_OFFSET : POSITIVE := 205;
   begin
```

```
    for WEEK in TIMETABLE_TYPES_PACKAGE.
                WEEK_NUMBER_TYPE
    loop

      if WEEKS_REQUIRED(WEEK) = TRUE then

        PERIOD_QUERY_STRING
        (PERIOD_1_QUERY_OFFSET .. PERIOD_1_QUERY_OFFSET + 4 )
        := "o    ";

        PERIOD_QUERY_STRING
        (PERIOD_2_QUERY_OFFSET .. PERIOD_2_QUERY_OFFSET + 4 )
        := "o    ";

        PERIOD_QUERY_STRING
        (PERIOD_3_QUERY_OFFSET .. PERIOD_3_QUERY_OFFSET + 4 )
        := "o    ";

      end if;
      PERIOD_1_QUERY_OFFSET := PERIOD_1_QUERY_OFFSET + 6;
      PERIOD_2_QUERY_OFFSET := PERIOD_2_QUERY_OFFSET + 6;
      PERIOD_3_QUERY_OFFSET := PERIOD_3_QUERY_OFFSET + 6;

    end loop;

  end;

  STRIP_SPACES
  (IN_STANDARD_STRING => PERIOD_QUERY_STRING,
   OUT_DYNAMIC_STRING => DYNAMIC_PERIOD_QUERY_STRING);

  -- Assert the query

  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (KB         => PERIOD_QUERY,
   IN_CLAUSE => SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                SUBSTRING_OF
                (THE_STRING => DYNAMIC_PERIOD_QUERY_STRING));

  -- Ask the query

  IT_PERIOD_INFERENCE_PACKAGE.
  SOLVE.
  START
  (THIS_QUERY => PERIOD_QUERY,
   THIS_KB    => PERIOD_KB);

  loop

    -- Get the result

    IT_PERIOD_INFERENCE_PACKAGE.
    CONTROL.
    GET_RESULT
    (OUT_LIST => VALUE_LIST);

    -- Check for No periods
```

```
if SYSTEM_TYPES_PACKAGE.
   DYNAMIC_STRING.
   IS_EQUAL
   (LEFT  => "No",
    RIGHT => SYSTEM_TYPES_PACKAGE.
             DYNAMIC_STRING_LIST_PACKAGE.
             ITEM_AT
             (LIST        => VALUE_LIST,
              NODE_NUMBER => 1)) then

  raise PERIODS_NOT_AVAILABLE;

end if;

-- Analyse response

EXTRACT_THREE_PERIODS
(PERIOD_1       => PERIOD1,
 PERIOD_2       => PERIOD2,
 PERIOD_3       => PERIOD3,
 THIS_DAY       => DAY,
 OLD_P1_CLAUSE  => OLD_P1_STRING,
 OLD_P2_CLAUSE  => OLD_P2_STRING,
 OLD_P3_CLAUSE  => OLD_P3_STRING,
 NEW_P1_CLAUSE  => NEW_P1_STRING,
 NEW_P2_CLAUSE  => NEW_P2_STRING,
 NEW_P3_CLAUSE  => NEW_P3_STRING,
 VALUE_LIST     => VALUE_LIST);

-- Is this solution acceptable?

declare
  COMMON_QUERY_OFFSET : POSITIVE := 18;
begin

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      COMMON_QUERY_STRING(WEEK + COMMON_QUERY_OFFSET) := 'o';
    end if;
    COMMON_QUERY_OFFSET := COMMON_QUERY_OFFSET + 1;

  end loop;
end;

COMMON_QUERY_STRING(8 .. 10) := OLD_P1_STRING(8 .. 10);
COMMON_QUERY_STRING(12 .. 13) := OLD_P1_STRING(12 ..13);

if COMMON_MODULES_AVAILABLE
   (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR        => ACTIVITY_PTR) and then
   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST              => ACTIVITY_PTR.
                               STAFF_LIST,
   MODULE                   => ACTIVITY_PTR.
                               FACT,
```

```
 DAY                          => DAY,
 PERIOD                       => PERIOD1,
 WEEK_ARRAY                   => WEEKS_REQUIRED,
 STAFF_FRAME_BASE_RECORD      => STAFF_PACKAGE.
                                 STAFF_FRAME_BASE_RECORD) then

COMMON_QUERY_STRING(12 .. 13) := OLD_P2_STRING(12 .. 13);

if COMMON_MODULES_AVAILABLE
 (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
  ACTIVITY_PTR        => ACTIVITY_PTR) and then
 STAFF_PACKAGE.
 ALL_STAFF_FREE
 (STAFF_LIST                  => ACTIVITY_PTR.
                                 STAFF_LIST,
 MODULE                       => ACTIVITY_PTR.
                                 FACT,
 DAY                          => DAY,
 PERIOD                       => PERIOD2,
 WEEK_ARRAY                   => WEEKS_REQUIRED,
 STAFF_FRAME_BASE_RECORD      => STAFF_PACKAGE.
                                 STAFF_FRAME_BASE_RECORD) then

  COMMON_QUERY_STRING(12 .. 13) := OLD_P3_STRING(12 ..13);

  if COMMON_MODULES_AVAILABLE
    (PERIOD_QUERY_STRING => COMMON_QUERY_STRING,
    ACTIVITY_PTR         => ACTIVITY_PTR) and then
    STAFF_PACKAGE.
    ALL_STAFF_FREE
    (STAFF_LIST                 => ACTIVITY_PTR.
                                   STAFF_LIST,
    MODULE                      => ACTIVITY_PTR.
                                   FACT,
    DAY                         => DAY,
    PERIOD                      => PERIOD3,
    WEEK_ARRAY                  => WEEKS_REQUIRED,
    STAFF_FRAME_BASE_RECORD     => STAFF_PACKAGE.
                                   STAFF_FRAME_BASE_RECORD) then

    PERIOD_1 := PERIOD1;
    PERIOD_2 := PERIOD2;
    PERIOD_3 := PERIOD3;
    THIS_DAY := DAY;

    -- Release solve if no more

    IT_PERIOD_INFERENCE_PACKAGE.
    CONTROL.
    NO_MORE;

    -- Retract current clauseS

    IT_PERIOD_INFERENCE_PACKAGE.
    LOGIC_KB.
    RETRACT
    (CLAUSE => OLD_P1_STRING,
     KB     => PERIOD_KB);

    IT_PERIOD_INFERENCE_PACKAGE.
```

```
   LOGIC_KB.
   RETRACT
   (CLAUSE => OLD_P2_STRING,
    KB     => PERIOD_KB);

   IT_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   RETRACT
   (CLAUSE => OLD_P3_STRING,
    KB     => PERIOD_KB);

   -- Assert updated clause

if NEW_P1_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

   IT_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (IN_CLAUSE => NEW_P1_STRING,
    KB        => IT_PERIOD_KB);

else

   IT_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (IN_CLAUSE  => NEW_P1_STRING,
    KB         => IT_PERIOD_KB,
    AT_BACK_OF => FALSE);

end if;

if NEW_P2_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

   IT_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (IN_CLAUSE => NEW_P2_STRING,
    KB        => IT_PERIOD_KB);

else

   IT_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (IN_CLAUSE  => NEW_P2_STRING,
    KB         => IT_PERIOD_KB,
    AT_BACK_OF => FALSE);

end if;

if NEW_P3_STRING(18 .. 40) =
   "(x,x,x,x,x,x,x,x,x,x,x)" then

   IT_PERIOD_INFERENCE_PACKAGE.
   LOGIC_KB.
   ASSERT
   (IN_CLAUSE => NEW_P3_STRING,
```

```
    KB          => IT_PERIOD_KB);

else

  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  ASSERT
  (IN_CLAUSE   => NEW_P3_STRING,
   KB          => IT_PERIOD_KB,
   AT_BACK_OF  => FALSE);

end if;

  -- Do same for all common modules

  ADJUST_COMMON_MODULE_PERIODS
  (DAY             => OLD_P1_STRING(8 .. 10),
   PERIOD          => OLD_P1_STRING(12 .. 13),
   WEEKS_REQUIRED  => WEEKS_REQUIRED,
   ACTIVITY_PTR    => ACTIVITY_PTR);

  STAFF_PACKAGE.
  MAKE_ALL_STAFF_BUSY
  (STAFF_LIST              => ACTIVITY_PTR.
                              STAFF_LIST,
  MODULE                   => ACTIVITY_PTR.
                              FACT,
  DAY                      => DAY,
  PERIOD                   => PERIOD1,
  WEEK_ARRAY               => WEEKS_REQUIRED,
  STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                              STAFF_FRAME_BASE_RECORD);

  ADJUST_COMMON_MODULE_PERIODS
  (DAY             => OLD_P2_STRING(8 .. 10),
   PERIOD          => OLD_P2_STRING(12 .. 13),
   WEEKS_REQUIRED  => WEEKS_REQUIRED,
   ACTIVITY_PTR    => ACTIVITY_PTR);

  STAFF_PACKAGE.
  MAKE_ALL_STAFF_BUSY
  (STAFF_LIST              => ACTIVITY_PTR.
                              STAFF_LIST,
  MODULE                   => ACTIVITY_PTR.
                              FACT,
  DAY                      => DAY,
  PERIOD                   => PERIOD2,
  WEEK_ARRAY               => WEEKS_REQUIRED,
  STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                              STAFF_FRAME_BASE_RECORD);

  ADJUST_COMMON_MODULE_PERIODS
  (DAY             => OLD_P3_STRING(8 .. 10),
   PERIOD          => OLD_P3_STRING(12 .. 13),
   WEEKS_REQUIRED  => WEEKS_REQUIRED,
   ACTIVITY_PTR    => ACTIVITY_PTR);

  STAFF_PACKAGE.
  MAKE_ALL_STAFF_BUSY
  (STAFF_LIST              => ACTIVITY_PTR.
```

```
                                        STAFF_LIST,
            MODULE                   => ACTIVITY_PTR.
                                        FACT,
            DAY                      => DAY,
            PERIOD                   => PERIOD3,
            WEEK_ARRAY               => WEEKS_REQUIRED,
            STAFF_FRAME_BASE_RECORD  => STAFF_PACKAGE.
                                        STAFF_FRAME_BASE_RECORD);

            SELECTED := TRUE;

            exit;

        end if;

      end if;

      else

        IT_PERIOD_INFERENCE_PACKAGE.
        CONTROL.
        GET_MORE;

      end if;
    end loop;
  ------------------------------------------------------------------
  exception
    when PERIODS_NOT_AVAILABLE =>
      SELECTED := FALSE;
      TEXT_IO.PUT_LINE
      ("Exception PERIODS_NOT_AVALABLE in PERIOD_KS_PACKAGE " &
       "at SELECT_3_IT_PERIODS");
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in PERIOD_KS_PACKAGE " &
       "at SELECT_3_IT_PERIODS");
  ------------------------------------------------------------------
  end SELECT_3_IT_PERIODS;
  ------------------------------------------------------------------

------------------------------------------------------------------
end PERIOD_IT_PACKAGE;
------------------------------------------------------------------
```

Period KS

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_KS_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function  : This package provides the blackboard transformation
--             from the degree activities level to the day level by
--             alllocating periods to the degree activities
--
-------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE,
     PERIOD_ESE_PACKAGE,
     PERIOD_IT_PACKAGE;
use PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE,
     PERIOD_ESE_PACKAGE,
     PERIOD_IT_PACKAGE;
-------------------------------------------------------------------
package PERIOD_KS_PACKAGE is
-------------------------------------------------------------------

  procedure PROCESS_EVENT
  (ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD   : in out TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_TYPE);

-------------------------------------------------------------------
end PERIOD_KS_PACKAGE;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : PERIOD_KS_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 29 July 1992
-- Function : This package provides the blackboard transformation
--             from the degree activity level to the day level by
--             allocating periods
--             to the degree activities
--
-------------------------------------------------------------------
package body PERIOD_KS_PACKAGE is
-------------------------------------------------------------------
  use TEXT_IO;
  -----------------------------------------------------------------
  --
  -- Process_event allocates periods to the activities
  --
  -----------------------------------------------------------------
  procedure PROCESS_EVENT
  -----------------------------------------------------------------
  (ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   BLACKBOARD   : in out TIMETABLE_BLACKBOARD_PACKAGE.
                         TIMETABLE_BLACKBOARD.
                         BLACKBOARD_TYPE) is
  -----------------------------------------------------------------

    PERIOD_1 : TIMETABLE_TYPES_PACKAGE.
               PERIOD_NUMBER_TYPE := 1;
    PERIOD_2 : TIMETABLE_TYPES_PACKAGE.
               PERIOD_NUMBER_TYPE := 1;
    PERIOD_3 : TIMETABLE_TYPES_PACKAGE.
               PERIOD_NUMBER_TYPE := 1;

    DAY : TIMETABLE_TYPES_PACKAGE.
          DAY_TYPE;

    PERIOD_PREFERENCE : TIMETABLE_TYPES_PACKAGE.
                        PERIOD_NUMBER_TYPE;

    SELECTED_1 : BOOLEAN := FALSE;
    SELECTED_2 : BOOLEAN := FALSE;
    SELECTED_3 : BOOLEAN := FALSE;

    WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                     WEEK_ARRAY_TYPE;

    DEGREE : CHARACTER := SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          ITEM_OF
                          (THE_STRING      => ACTIVITY_PTR.
                                              FACT,
                                              AT_THE_POSITION => 1);

    DEGREE_ERROR : exception;
    NUMBER_OF_PERIODS_ERROR : exception;
    -----------------------------------------------------------------
    function CONVERT
```

```
    (PERIOD : in TIMETABLE_TYPES_PACKAGE.
                PERIOD_NUMBER_TYPE)
    return STANDARD.STRING is
    ------------------------------------------------------------
      CONVERTED_PERIOD : STANDARD.STRING(1 .. 1);

    begin

      CONVERTED_PERIOD := TIMETABLE_TYPES_PACKAGE.
                          PERIOD_NUMBER_TYPE'
                          IMAGE
                          (PERIOD)
                          (TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE'
                           IMAGE
                           (PERIOD)'LAST ..
                           TIMETABLE_TYPES_PACKAGE.
                           PERIOD_NUMBER_TYPE'
                           IMAGE
                           (PERIOD)'LAST);

      -- Returning the expression causes a constraint error in
      -- Meridian Ada!
      -- Hence the nead for the local variable and assignment.

      return CONVERTED_PERIOD;
  ------------------------------------------------------------
    end CONVERT;
  ------------------------------------------------------------
  begin

    -- Establish the weeks that the period is required

    if ACTIVITY_PTR.ACTIVITY = 'C' then

      null; -- A carousel activity

    else

      CONVERT
      (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     SUBSTRING_OF
                     (THE_STRING => ACTIVITY_PTR.
                                    FREQUENCY),
       WEEK_ARRAY => WEEKS_REQUIRED);

      if ACTIVITY_PTR.HAS_PREFERENCE then

        if PREFERED_ESE_PERIOD_OCCUPIED
           (WEEKS_REQUIRED => WEEKS_REQUIRED,
            ESE_PERIOD_KB  => ESE_PERIOD_KB,
            ACTIVITY_PTR   => ACTIVITY_PTR) then

          declare
            ALREADY_PREFERED : exception;
          begin

            if OCCUPIED_PERIOD_ALREADY_PREFERED
               (DAY         => SYSTEM_TYPES_PACKAGE.
```

```
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       DAY_PREFERENCE),
          PERIOD      => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       PERIOD_PREFERENCE),
          BLACKBOARD =>  BLACKBOARD) then

        raise ALREADY_PREFERED;

      else

         RESET_KNOWLEDGE_BASES
         (DAY         => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       DAY_PREFERENCE),
          PERIOD      => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       PERIOD_PREFERENCE),
          BLACKBOARD => BLACKBOARD);

         RESCHEDULE_OCCUPIED_PERIOD_ACTIVITIES
         (PERIOD      => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       PERIOD_PREFERENCE),
          DAY         => SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       DAY_PREFERENCE),
          BLACKBOARD => BLACKBOARD);

      end if;
    end;
  end if;

  PERIOD_PREFERENCE := TIMETABLE_TYPES_PACKAGE.
                       PERIOD_NUMBER_TYPE'
                       VALUE
                       (SYSTEM_TYPES_PACKAGE.
                        DYNAMIC_STRING.
                        SUBSTRING_OF
                        (THE_STRING => ACTIVITY_PTR.
                                       PERIOD_PREFERENCE));

  case ACTIVITY_PTR.NUMBER_OF_PERIODS is

    when 1       => RESERVE_1_PREFERED_ESE_PERIOD
                    (PERIOD          =>  CONVERT
                                        (PERIOD =>
                                        PERIOD_PREFERENCE),
```

```
                    THIS_PERIOD     => PERIOD_1,
                    DAY             => SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       SUBSTRING_OF
                                       (THE_STRING =>
                                       ACTIVITY_PTR.
                                       DAY_PREFERENCE),
                    THIS_DAY        => DAY,
                    WEEKS_REQUIRED  => WEEKS_REQUIRED,
                    PERIOD_KB       => ESE_PERIOD_KB,
                    ACTIVITY_PTR    => ACTIVITY_PTR,
                    BLACKBOARD      => BLACKBOARD,
                    SELECTED        => SELECTED_1);

when 2      => RESERVE_1_PREFERED_ESE_PERIOD
                    (PERIOD         =>  CONVERT
                                       (PERIOD =>
                                        PERIOD_PREFERENCE),
                    THIS_PERIOD     => PERIOD_1,
                    DAY             => SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       SUBSTRING_OF
                                       (THE_STRING =>
                                        ACTIVITY_PTR.
                                        DAY_PREFERENCE),
                    THIS_DAY        => DAY,
                    WEEKS_REQUIRED  => WEEKS_REQUIRED,
                    PERIOD_KB       => ESE_PERIOD_KB,
                    ACTIVITY_PTR    => ACTIVITY_PTR,
                    BLACKBOARD      => BLACKBOARD,
                    SELECTED        => SELECTED_2);

                   RESERVE_1_PREFERED_ESE_PERIOD
                    (PERIOD         =>  CONVERT
                                       (PERIOD =>
                                        PERIOD_PREFERENCE + 1),
                    THIS_PERIOD     => PERIOD_2,
                    DAY             => SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       SUBSTRING_OF
                                       (THE_STRING =>
                                       ACTIVITY_PTR.
                                       DAY_PREFERENCE),
                    THIS_DAY        => DAY,
                    WEEKS_REQUIRED  => WEEKS_REQUIRED,
                    PERIOD_KB       => ESE_PERIOD_KB,
                    ACTIVITY_PTR    => ACTIVITY_PTR,
                    BLACKBOARD      => BLACKBOARD,
                    SELECTED        => SELECTED_2);

when 3      => RESERVE_1_PREFERED_ESE_PERIOD
                    (PERIOD         =>  CONVERT
                                       (PERIOD =>
                                        PERIOD_PREFERENCE),
                    THIS_PERIOD     => PERIOD_1,
                    DAY             => SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       SUBSTRING_OF
                                       (THE_STRING =>
                                       ACTIVITY_PTR
```

```
                                        .DAY_PREFERENCE),
                    THIS_DAY        => DAY,
                    WEEKS_REQUIRED  => WEEKS_REQUIRED,
                    PERIOD_KB       => ESE_PERIOD_KB,
                    ACTIVITY_PTR    => ACTIVITY_PTR,
                    BLACKBOARD      => BLACKBOARD,
                    SELECTED        => SELECTED_3);

                  RESERVE_1_PREFERED_ESE_PERIOD
                  (PERIOD           =>  CONVERT
                                        (PERIOD =>
                                         PERIOD_PREFERENCE + 1),
                    THIS_PERIOD     => PERIOD_2,
                    DAY             => SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       SUBSTRING_OF
                                       (THE_STRING =>
                                        ACTIVITY_PTR.
                                        DAY_PREFERENCE),
                    THIS_DAY        => DAY,
                    WEEKS_REQUIRED  => WEEKS_REQUIRED,
                    PERIOD_KB       => ESE_PERIOD_KB,
                    ACTIVITY_PTR    => ACTIVITY_PTR,
                    BLACKBOARD      => BLACKBOARD,
                    SELECTED        => SELECTED_3);

                  RESERVE_1_PREFERED_ESE_PERIOD
                  (PERIOD           =>  CONVERT
                                        (PERIOD =>
                                         PERIOD_PREFERENCE + 2),
                    THIS_PERIOD     => PERIOD_3,
                    DAY             => SYSTEM_TYPES_PACKAGE.
                                       DYNAMIC_STRING.
                                       SUBSTRING_OF
                                       (THE_STRING =>
                                        ACTIVITY_PTR
                                       .DAY_PREFERENCE),
                    THIS_DAY        => DAY,
                    WEEKS_REQUIRED  => WEEKS_REQUIRED,
                    PERIOD_KB       => ESE_PERIOD_KB,
                    ACTIVITY_PTR    => ACTIVITY_PTR,
                    BLACKBOARD      => BLACKBOARD,
                    SELECTED        => SELECTED_3);

    when others => raise NUMBER_OF_PERIODS_ERROR;

  end case;

else

-- Select_period from appropriate degree period kb

case ACTIVITY_PTR.
     NUMBER_OF_PERIODS is
  when 1 => case DEGREE is
              when 'E' => SELECT_1_ESE_PERIOD
                          (PERIOD_1       => PERIOD_1,
                           THIS_DAY       => DAY,
                           WEEKS_REQUIRED => WEEKS_REQUIRED,
                           PERIOD_KB      => ESE_PERIOD_KB,
```

```
                        ACTIVITY_PTR    => ACTIVITY_PTR,
                        SELECTED        => SELECTED_1);

            when 'I' => SELECT_1_IT_PERIOD
                        (PERIOD_1       => PERIOD_1,
                         THIS_DAY       => DAY,
                         WEEKS_REQUIRED => WEEKS_REQUIRED,
                         PERIOD_KB      => IT_PERIOD_KB,
                         ACTIVITY_PTR   => ACTIVITY_PTR,
                         SELECTED       => SELECTED_1);

             when others => raise DEGREE_ERROR;

          end case;

          ADD_PERIOD_DETAILS
          (THIS_DAY       => DAY,
           FIRST_PERIOD   => PERIOD_1,
           LAST_PERIOD    => PERIOD_1,
           WEEKS_REQUIRED => WEEKS_REQUIRED,
           ACTIVITY_PTR   => ACTIVITY_PTR,
           BLACKBOARD     => BLACKBOARD);

when 2 => case DEGREE is
            when 'E' => SELECT_2_ESE_PERIODS
                        (PERIOD_1       => PERIOD_1,
                         PERIOD_2       => PERIOD_2,
                         THIS_DAY       => DAY,
                         WEEKS_REQUIRED => WEEKS_REQUIRED,
                         PERIOD_KB      => ESE_PERIOD_KB,
                         ACTIVITY_PTR   => ACTIVITY_PTR,
                         SELECTED       => SELECTED_2);

            when 'I' => SELECT_2_IT_PERIODS
                        (PERIOD_1       => PERIOD_1,
                         PERIOD_2       => PERIOD_2,
                         THIS_DAY       => DAY,
                         WEEKS_REQUIRED => WEEKS_REQUIRED,
                         PERIOD_KB      => IT_PERIOD_KB,
                         ACTIVITY_PTR   => ACTIVITY_PTR,
                         SELECTED       => SELECTED_2);

            when others => raise DEGREE_ERROR;

           end case;

          ADD_PERIOD_DETAILS
          (THIS_DAY       => DAY,
           FIRST_PERIOD   => PERIOD_1,
           LAST_PERIOD    => PERIOD_2,
           WEEKS_REQUIRED => WEEKS_REQUIRED,
           ACTIVITY_PTR   => ACTIVITY_PTR,
           BLACKBOARD     => BLACKBOARD);

when 3 =>case DEGREE is
            when 'E' => SELECT_3_ESE_PERIODS
                        (PERIOD_1       => PERIOD_1,
                         PERIOD_2       => PERIOD_2,
                         PERIOD_3       => PERIOD_3,
                         THIS_DAY       => DAY,
```

```
                                    WEEKS_REQUIRED => WEEKS_REQUIRED,
                                    PERIOD_KB      => ESE_PERIOD_KB,
                                    ACTIVITY_PTR   => ACTIVITY_PTR,
                                    SELECTED       => SELECTED_3);

                        when 'I' => SELECT_3_IT_PERIODS
                                    (PERIOD_1       => PERIOD_1,
                                     PERIOD_2       => PERIOD_2,
                                     PERIOD_3       => PERIOD_3,
                                     THIS_DAY       => DAY,
                                     WEEKS_REQUIRED => WEEKS_REQUIRED,
                                     PERIOD_KB      => IT_PERIOD_KB,
                                     ACTIVITY_PTR   => ACTIVITY_PTR,
                                     SELECTED       => SELECTED_3);

                          when others => raise DEGREE_ERROR;

                       end case;

                       ADD_PERIOD_DETAILS
                       (THIS_DAY       => DAY,
                        FIRST_PERIOD   => PERIOD_1,
                        LAST_PERIOD    => PERIOD_3,
                        WEEKS_REQUIRED => WEEKS_REQUIRED,
                        ACTIVITY_PTR   => ACTIVITY_PTR,
                        BLACKBOARD     => BLACKBOARD);

              when others => raise NUMBER_OF_PERIODS_ERROR;
            end case;

          end if;

        end if;

        TEXT_IO.NEW_LINE;

      ------------------------------------------------------------------
      exception
        when DEGREE_ERROR =>
          TEXT_IO.PUT_LINE
          ("Exception DEGREE_ERROR in PERIOD_KS_PACKAGE at " &
           "PROCESS_EVENT");
        when NUMBER_OF_PERIODS_ERROR =>
          TEXT_IO.PUT_LINE
          ("Exception NUMBER_OF_PERIODS_ERROR in PERIOD_KS_PACKAGE" &
           " at PROCESS_EVENT");
        when OTHERS =>
          TEXT_IO.PUT_LINE
          ("Exception raised in PERIOD_KS_PACKAGE at " &
           "PROCESS_EVENT");
      ------------------------------------------------------------------
      end PROCESS_EVENT;
      ------------------------------------------------------------------
    begin

      ESE_PERIOD_INFERENCE_PACKAGE.
      LOGIC_KB.
      BUILD
      (KB => ESE_PERIOD_KB,
       FILE_NAME => "ESEperiod.pro");
```

```
  IT_PERIOD_INFERENCE_PACKAGE.
  LOGIC_KB.
  BUILD
  (KB => IT_PERIOD_KB,
   FILE_NAME => "ITperiod.pro");

-------------------------------------------------------------------------
end PERIOD_KS_PACKAGE;
-------------------------------------------------------------------------
```

Timetable KS

```
------------------------------------------------------------------
--
-- Unit      : TIMETABLE_KS_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 24 September 1992
-- Function  : This package provides the blackboard transformation to
--             turn the top level of the blackboard into the
--             timetable
--
------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE;
------------------------------------------------------------------
package TIMETABLE_KS_PACKAGE is
------------------------------------------------------------------

  type DEGREE_TYPE is (IT_DEGREE, ESE_DEGREE);

  subtype DAYS_IN_WEEK is TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_ITEM_TYPE range
                          TIMETABLE_TYPES_PACKAGE.MONDAY ..
                          TIMETABLE_TYPES_PACKAGE.FRIDAY;

  type GROUP_TYPE is array
                     (1..20) of
                     SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     STRING;

  type CONSOLIDATED_PERIOD_TYPE is array(1 .. 10,
                                         DAYS_IN_WEEK,
                                         DEGREE_TYPE) of GROUP_TYPE;

  type CONSOLIDATED_BUFFER_TYPE is array(TIMETABLE_TYPES_PACKAGE.
                                         PERIOD_NUMBER_TYPE) of
                                         CONSOLIDATED_PERIOD_TYPE;

  CONSOLIDATED_BUFFER : CONSOLIDATED_BUFFER_TYPE;

  procedure PROCESS_EVENT
  (BLACKBOARD    : in TIMETABLE_BLACKBOARD_PACKAGE.
                      TIMETABLE_BLACKBOARD.
                      BLACKBOARD_TYPE);

  procedure GET_BUFFER
  (BUFFER :     out CONSOLIDATED_BUFFER_TYPE);

------------------------------------------------------------------
end TIMETABLE_KS_PACKAGE;
------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : TIMETABLE_KS_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 24 September 1992
-- Function  : This package provides the blackboard transformation to
--             turn the top level of the blackboard into the
--             timetable
--
-------------------------------------------------------------------
package body TIMETABLE_KS_PACKAGE is
-------------------------------------------------------------------

  use TEXT_IO;
  use TIMETABLE_TYPES_PACKAGE;

  package PERIOD_IO is new TEXT_IO.INTEGER_IO
                           (TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE);

  package INTEGER_IO is new TEXT_IO.INTEGER_IO(INTEGER);

  IT                : TEXT_IO.FILE_TYPE;
  ESE               : TEXT_IO.FILE_TYPE;

  -----------------------------------------------------------------
  procedure ADD_TIMETABLE_ENTRIES
  -----------------------------------------------------------------
  (CONSOLIDATED_PERIOD          : in out CONSOLIDATED_PERIOD_TYPE;
   DAY                          : in     DAYS_IN_WEEK;
   PERIOD_MODULE_LIST           : in out SYSTEM_TYPES_PACKAGE.
                                         DYNAMIC_STRING_LIST_PACKAGE.
                                         LIST_TYPE;
   PERIOD_DETAILS_PTR_ARRAY     : in     TIMETABLE_TYPES_PACKAGE.
                                         PERIOD_FRAME_BASE_PACKAGE.
                                         FACET_PACKAGE.
                                         FACET_TREE_PACKAGE.
                                         ITEM_ARRAY_TYPE;
   NUMBER_OF_DETAILS            : in     NATURAL;
   ESE_GROUP                    : in out POSITIVE;
   IT_GROUP                     : in out POSITIVE) is
  -----------------------------------------------------------------

    MODULE_NAME : SYSTEM_TYPES_PACKAGE.
                  DYNAMIC_STRING.
                  STRING;

    MODULE_ERROR : exception;
    DEGREE_ERROR : exception;

    procedure ADD_GROUP_TO
    (CONSOLIDATED_PERIOD : in out CONSOLIDATED_PERIOD_TYPE;
     DAY                 : in     DAYS_IN_WEEK;
     DEGREE              : in     DEGREE_TYPE;
     GROUP               : in out POSITIVE) is

      ROW : POSITIVE := 1;

    begin
```

```
-- copy prime module to consolidated period

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_STRING => MODULE_NAME,
 TO_THE_STRING   => CONSOLIDATED_PERIOD
                    (GROUP, DAY, DEGREE)(ROW));

for COMMON_MODULE_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING_LIST_PACKAGE.
                                 LENGTH_OF
                                 (LIST => PERIOD_MODULE_LIST)
loop

  -- Copy all common modules to consolidated period

  ROW := ROW + 1;

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_STRING => SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING_LIST_PACKAGE.
                      ITEM_AT
                      (LIST        => PERIOD_MODULE_LIST,
                       NODE_NUMBER => COMMON_MODULE_NUMBER),
   TO_THE_STRING   => CONSOLIDATED_PERIOD
                      (GROUP, DAY, DEGREE)(ROW));

end loop;

for PERIOD_DETAIL_NUMBER in 1 .. NUMBER_OF_DETAILS
loop

  ROW := ROW + 1;

  -- copy all details to consolidated period

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_STRING => PERIOD_DETAILS_PTR_ARRAY
                      (PERIOD_DETAIL_NUMBER).
                      INITIALS,
   TO_THE_STRING   => CONSOLIDATED_PERIOD
                      (GROUP, DAY, DEGREE)(ROW));

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  APPEND
  (THE_SUBSTRING => " ",
   TO_THE_STRING => CONSOLIDATED_PERIOD
                    (GROUP, DAY, DEGREE)(ROW));

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  APPEND
  (THE_STRING    => PERIOD_DETAILS_PTR_ARRAY
```

```
                        (PERIOD_DETAIL_NUMBER).
                        WEEK_CODE,
        TO_THE_STRING => CONSOLIDATED_PERIOD
                        (GROUP, DAY, DEGREE)(ROW));

      ROW := ROW + 1;

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING.
      COPY
      (FROM_THE_STRING => PERIOD_DETAILS_PTR_ARRAY
                          (PERIOD_DETAIL_NUMBER).
                          ROOM,
       TO_THE_STRING   => CONSOLIDATED_PERIOD
                          (GROUP, DAY, DEGREE)(ROW));

    end loop;
  ------------------------------------------------------------------
  exception
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in TIMETABLE_KS_PACKAGE at " &
       "ADD_GROUP_TO");
  ------------------------------------------------------------------
  end ADD_GROUP_TO;
----------------------------------------------------------------------
begin
----------------------------------------------------------------------

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => PERIOD_MODULE_LIST,
     THIS_ITEM => MODULE_NAME);

    declare

      PRIMARY_MODULE : CHARACTER := SYSTEM_TYPES_PACKAGE.
                                    DYNAMIC_STRING.
                                    ITEM_OF
                                    (THE_STRING => MODULE_NAME,
                                     AT_THE_POSITION => 1);
    begin

      case SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           ITEM_OF
           (THE_STRING => MODULE_NAME,
            AT_THE_POSITION => 1) is

        when 'E' => ADD_GROUP_TO
                    (CONSOLIDATED_PERIOD => CONSOLIDATED_PERIOD,
                     DAY                 => DAY,
                     DEGREE              => ESE_DEGREE,
                     GROUP               => ESE_GROUP);

                     ESE_GROUP := ESE_GROUP + 1;

        when 'I' => ADD_GROUP_TO
                    (CONSOLIDATED_PERIOD => CONSOLIDATED_PERIOD,
```

```
                DAY                    => DAY,
                DEGREE                 => IT_DEGREE,
                GROUP                  => IT_GROUP);

              IT_GROUP:= IT_GROUP + 1;

  when others => raise DEGREE_ERROR;

 end case;

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING_LIST_PACKAGE.
PUT_ON_BACK_OF
(LIST      => PERIOD_MODULE_LIST,
 THIS_ITEM => MODULE_NAME);

for MODULE in 1 .. SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING_LIST_PACKAGE.
                   LENGTH_OF
                   (LIST => PERIOD_MODULE_LIST) - 1
loop

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING_LIST_PACKAGE.
  GET_FROM_FRONT_OF
  (LIST      => PERIOD_MODULE_LIST,
   THIS_ITEM => MODULE_NAME);

  if not (PRIMARY_MODULE = SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           ITEM_OF
                           (THE_STRING => MODULE_NAME,
                            AT_THE_POSITION => 1)) then

    case SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         ITEM_OF
         (THE_STRING => MODULE_NAME,
          AT_THE_POSITION => 1) is

      when 'E' => ADD_GROUP_TO
                  (CONSOLIDATED_PERIOD =>
                   CONSOLIDATED_PERIOD,
                   DAY                   => DAY,
                   DEGREE                => ESE_DEGREE,
                   GROUP                 => ESE_GROUP);

                  ESE_GROUP := ESE_GROUP + 1;

      when 'I' => ADD_GROUP_TO
                  (CONSOLIDATED_PERIOD =>
                   CONSOLIDATED_PERIOD,
                   DAY                   => DAY,
                   DEGREE                => IT_DEGREE,
                   GROUP                 => IT_GROUP);

                  IT_GROUP := IT_GROUP + 1;

      when others => raise DEGREE_ERROR;
```

```
          end case;

        end if;

      SYSTEM_TYPES_PACKAGE.
      DYNAMIC_STRING_LIST_PACKAGE.
      PUT_ON_BACK_OF
      (LIST      => PERIOD_MODULE_LIST,
       THIS_ITEM => MODULE_NAME);

    end loop;
  end;
-------------------------------------------------------------------
exception

  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR in TIMETABLE_KS_PACKAGE at " &
     "ADD_TIMETABLE_ENTRIES");
  when MODULE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception MODULE_ERROR in TIMETABLE_KS_PACKAGE at " &
                      "ADD_TIMETABLE_ENTRIES");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in TIMETABLE_KS_PACKAGE at " &
     "ADD_TIMETABLE_ENTRIES");
-------------------------------------------------------------------
end ADD_TIMETABLE_ENTRIES;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PRODUCE_TIMETABLE
-------------------------------------------------------------------
(BLACKBOARD : in TIMETABLE_BLACKBOARD_PACKAGE.
                TIMETABLE_BLACKBOARD.
                BLACKBOARD_TYPE) is
-------------------------------------------------------------------

  ROOT                      : SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;
  RESOURCE                  : SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              STRING;

  DASHES       : STRING(1 .. 80) := (1 .. 80 => '-');

  DAY_ERROR : exception;
  PERIOD_ERROR : exception;

  -----------------------------------------------------------------
  procedure ADD_TO_FILE
  -----------------------------------------------------------------
  (CONSOLIDATED_PERIOD : in CONSOLIDATED_PERIOD_TYPE) is
  -----------------------------------------------------------------
    COLUMN     : TEXT_IO.POSITIVE_COUNT := 3;
    OUTPUT     : BOOLEAN := FALSE;
```

```
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  for DEGREE in DEGREE_TYPE
  loop

  for GROUP in 1 .. 10
  loop

    for ROW in 1 .. 20
    loop

      OUTPUT := FALSE;

      for DAY in DAYS_IN_WEEK
      loop
        if not SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               IS_NULL
               (THE_STRING      => CONSOLIDATED_PERIOD
                                   (GROUP, DAY, DEGREE)(1)) then

         -- If present then file detail

         if not SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                IS_NULL
                (THE_STRING => CONSOLIDATED_PERIOD
                               (GROUP, DAY, DEGREE)(ROW)) then

           OUTPUT := TRUE;

           if DEGREE = ESE_DEGREE then
             TEXT_IO.SET_COL(FILE => ESE, TO => COLUMN);
           else
             TEXT_IO.SET_COL(FILE => IT,  TO => COLUMN);
           end if;

           case DEGREE is

             when ESE_DEGREE => TEXT_IO.
                                PUT
                                (FILE => ESE,
                                 ITEM => SYSTEM_TYPES_PACKAGE.
                                         DYNAMIC_STRING.
                                         SUBSTRING_OF
                                         (THE_STRING =>
                                          CONSOLIDATED_PERIOD
                                          (GROUP, DAY, DEGREE)
                                          (ROW)));

             when IT_DEGREE => TEXT_IO.
                               PUT
                               (FILE => IT,
                                ITEM => SYSTEM_TYPES_PACKAGE.
                                        DYNAMIC_STRING.
                                        SUBSTRING_OF
                                        (THE_STRING =>
                                         CONSOLIDATED_PERIOD
```

```
                                                    (GROUP, DAY, DEGREE)
                                                    (ROW)));

              when others => raise DEGREE_ERROR;

            end case;

          end if;

        end if;

        if DAY /= DAYS_IN_WEEK'LAST then
          COLUMN := COLUMN + 16;
        end if;

      end loop; -- Day

      if OUTPUT then

        if DEGREE = ESE_DEGREE then

          TEXT_IO.NEW_LINE(FILE => ESE);

        else

          TEXT_IO.NEW_LINE(FILE => IT);

        end if;

      end if;

      COLUMN := 3;
    end loop; -- Row
    end loop; -- Group

    if DEGREE = ESE_DEGREE then

      TEXT_IO.PUT_LINE(FILE => ESE, ITEM => DASHES);

    else

      TEXT_IO.PUT_LINE(FILE => IT, ITEM => DASHES);

    end if;

  end loop; --Degree
  ------------------------------------------------------------------
  exception
    when DEGREE_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception DEGREE_ERROR in TIMETABLE_KS_PACKAGE at " &
       "ADD_TO_FILE");
  ------------------------------------------------------------------
  end ADD_TO_FILE;
  ------------------------------------------------------------------

------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
```

```
DYNAMIC_STRING.
COPY
(FROM_THE_SUBSTRING => "RESOURCE",
 TO_THE_STRING      => RESOURCE);

SYSTEM_TYPES_PACKAGE.
DYNAMIC_STRING.
COPY
(FROM_THE_SUBSTRING => "ROOT",
 TO_THE_STRING      => ROOT);

TEXT_IO.OPEN(FILE => IT,
             MODE => TEXT_IO.OUT_FILE,
             NAME => "IT.out");

TEXT_IO.OPEN(FILE => ESE,
             MODE => TEXT_IO.OUT_FILE,
             NAME => "ESE.out");

TEXT_IO.PUT_LINE(FILE => ESE, ITEM => DASHES);
TEXT_IO.PUT_LINE(FILE => IT,  ITEM => DASHES);

for DAY in DAYS_IN_WEEK
loop

  case DAY is
  when TIMETABLE_TYPES_PACKAGE.
       MONDAY    =>
       TEXT_IO.SET_COL(FILE => IT,  TO => 3);
       TEXT_IO.SET_COL(FILE => ESE, TO => 3);
  when TIMETABLE_TYPES_PACKAGE.
       TUESDAY   =>
       TEXT_IO.SET_COL(FILE => IT,  TO => 19);
       TEXT_IO.SET_COL(FILE => ESE, TO => 19);
  when TIMETABLE_TYPES_PACKAGE.
       WEDNESDAY =>
       TEXT_IO.SET_COL(FILE => IT,  TO => 35);
       TEXT_IO.SET_COL(FILE => ESE, TO => 35);
  when TIMETABLE_TYPES_PACKAGE.
       THURSDAY  =>
       TEXT_IO.SET_COL(FILE => IT,  TO => 51);
       TEXT_IO.SET_COL(FILE => ESE, TO => 51);
  when TIMETABLE_TYPES_PACKAGE.
       FRIDAY    =>
       TEXT_IO.SET_COL(FILE => IT,  TO => 67);
       TEXT_IO.SET_COL(FILE => ESE, TO => 67);
  when others    => raise DAY_ERROR;
  end case;

  TEXT_IO.PUT
  (FILE => ESE, ITEM => DAYS_IN_WEEK'IMAGE(DAY));

  TEXT_IO.PUT
  (FILE => IT, ITEM => DAYS_IN_WEEK'IMAGE(DAY));

end loop;

TEXT_IO.NEW_LINE(FILE => ESE);
TEXT_IO.NEW_LINE(FILE => IT);
```

```
TEXT_IO.PUT_LINE(FILE => ESE, ITEM => DASHES);
TEXT_IO.PUT_LINE(FILE => IT,  ITEM => DASHES);

for PERIOD_NUMBER in TIMETABLE_TYPES_PACKAGE.
                     PERIOD_NUMBER_TYPE
loop

  PERIOD_IO.PUT(FILE => ESE, ITEM => PERIOD_NUMBER, WIDTH => 1);
  PERIOD_IO.PUT(FILE => IT,  ITEM => PERIOD_NUMBER, WIDTH => 1);

  declare

    CONSOLIDATED_PERIOD     : CONSOLIDATED_PERIOD_TYPE;

  begin

    for DAY in DAYS_IN_WEEK
    loop
      declare

        PERIOD_ENTRIES_PTR_LIST  : TIMETABLE_TYPES_PACKAGE.
        PERIOD_FRAME_BASE_PACKAGE.
        FRAME_PACKAGE.
        LIST_PACKAGE.
        LIST_TYPE;

        ESE_GROUP : POSITIVE := 1;
        IT_GROUP  : POSITIVE := 1;

      begin

        if not TIMETABLE_TYPES_PACKAGE.
               PERIOD_FRAME_BASE_PACKAGE.
               IS_EMPTY
               (TIMETABLE_BLACKBOARD_PACKAGE.
                TIMETABLE_BLACKBOARD.
                BLACKBOARD_ITEM
                (BLACKBOARD  => TIMETABLE_BLACKBOARD_PACKAGE.
                                BLACKBOARD,
                 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                DAYS,
                 ITEM_INDEX  => DAY).
                 PERIOD(PERIOD_NUMBER).
                 PERIOD_DETAIL_FRAMES) then

          TIMETABLE_TYPES_PACKAGE.
          PERIOD_FRAME_BASE_PACKAGE.
          GET
          (SUBFRAME_LIST      => PERIOD_ENTRIES_PTR_LIST,
           FRAME_NAME         => ROOT,
           FRAME_BASE_RECORD => TIMETABLE_BLACKBOARD_PACKAGE.
                                TIMETABLE_BLACKBOARD.
                                BLACKBOARD_ITEM
                                (BLACKBOARD  =>
                                 TIMETABLE_BLACKBOARD_PACKAGE.
                                 BLACKBOARD,
                                 LEVEL_INDEX =>
                                 TIMETABLE_TYPES_PACKAGE.
                                 DAYS,
                                 ITEM_INDEX  => DAY).
```

```
                              PERIOD(PERIOD_NUMBER).
                              PERIOD_DETAIL_FRAMES);

    for PERIOD_GROUP in 1 .. TIMETABLE_TYPES_PACKAGE.
                              PERIOD_FRAME_BASE_PACKAGE.
                              FRAME_PACKAGE.
                              LIST_PACKAGE.
                              LENGTH_OF
                              (LIST =>
                               PERIOD_ENTRIES_PTR_LIST)
    loop
      declare

        PERIOD_MODULE_LIST :
        SYSTEM_TYPES_PACKAGE.
        DYNAMIC_STRING_LIST_PACKAGE.
        LIST_TYPE;

        PERIOD_DETAILS_PTR_ARRAY :
        TIMETABLE_TYPES_PACKAGE.
        PERIOD_FRAME_BASE_PACKAGE.
        FACET_PACKAGE.
        FACET_TREE_PACKAGE.
        ITEM_ARRAY_TYPE(1 ..10);

        NUMBER_OF_DETAILS : NATURAL := 0;

      begin

        NUMBER_OF_DETAILS := 0;

        TIMETABLE_TYPES_PACKAGE.
        PERIOD_FRAME_BASE_PACKAGE.
        GET
        (SUBFRAME_LIST     => PERIOD_MODULE_LIST,
         FACET_PTR_ARRAY   => PERIOD_DETAILS_PTR_ARRAY,
         NUMBER_OF_FACETS  => NUMBER_OF_DETAILS,
         SLOT_NAME         => RESOURCE,
         FRAME_PTR         => TIMETABLE_TYPES_PACKAGE.
                              PERIOD_FRAME_BASE_PACKAGE.
                              FRAME_PACKAGE.
                              LIST_PACKAGE.
                              ITEM_AT
                              (LIST        =>
                               PERIOD_ENTRIES_PTR_LIST,
                               NODE_NUMBER =>
                               PERIOD_GROUP));

        ADD_TIMETABLE_ENTRIES
        (CONSOLIDATED_PERIOD      => CONSOLIDATED_PERIOD,
         DAY                      => DAY,
         PERIOD_MODULE_LIST       => PERIOD_MODULE_LIST,
         PERIOD_DETAILS_PTR_ARRAY =>
         PERIOD_DETAILS_PTR_ARRAY,
         NUMBER_OF_DETAILS        => NUMBER_OF_DETAILS,
         ESE_GROUP                => ESE_GROUP,
         IT_GROUP                 => IT_GROUP);

        -- File details
```

```
                end;

              end loop;

            end if;

          end;

        end loop; -- Day

        ADD_TO_FILE
        (CONSOLIDATED_PERIOD => CONSOLIDATED_PERIOD);

        CONSOLIDATED_BUFFER(PERIOD_NUMBER) := CONSOLIDATED_PERIOD;

      end;
    end loop; -- Period

    TEXT_IO.CLOSE(FILE => IT);
    TEXT_IO.CLOSE(FILE => ESE);
  ------------------------------------------------------------------
    exception
      when OTHERS =>
        TEXT_IO.PUT_LINE
        ("Exception OTHERS in TIMETABLE_KS_PACKAGE at " &
         "PRODUCE_TIMETABLE");
  ------------------------------------------------------------------
  end PRODUCE_TIMETABLE;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure PROCESS_EVENT
  ------------------------------------------------------------------
  (BLACKBOARD    : in TIMETABLE_BLACKBOARD_PACKAGE.
                      TIMETABLE_BLACKBOARD.
                      BLACKBOARD_TYPE) is
  ------------------------------------------------------------------
  begin
    PRODUCE_TIMETABLE(BLACKBOARD => TIMETABLE_BLACKBOARD_PACKAGE.
                                    BLACKBOARD);
  ------------------------------------------------------------------
  end PROCESS_EVENT;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure GET_BUFFER
  ------------------------------------------------------------------
  (BUFFER :    out CONSOLIDATED_BUFFER_TYPE) is
  ------------------------------------------------------------------
  begin
   BUFFER := CONSOLIDATED_BUFFER;
  ------------------------------------------------------------------
  end GET_BUFFER;
  ------------------------------------------------------------------

------------------------------------------------------------------
end TIMETABLE_KS_PACKAGE;
------------------------------------------------------------------
```

Timetable Scheduler

```
------------------------------------------------------------------------
--
-- Unit      : TIMETABLE_SCHEDULER subprogram unit
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 1 January 1992
-- Function : Controls the timetable production
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     BLACKBOARD_INITIALISE_KS_PACKAGE,
     SYLLABUS_KS_PACKAGE,
     REQUIREMENT_KS_PACKAGE,
     COMMON_KS_PACKAGE,
     ACTIVITY_KS_PACKAGE,
     STAFF_KS_PACKAGE,
     PERIOD_KS_PACKAGE,
     TIMETABLE_KS_PACKAGE;
------------------------------------------------------------------------
procedure TIMETABLE_SCHEDULER is
------------------------------------------------------------------------

  package DAY_IO is new TEXT_IO.
                        ENUMERATION_IO
                        (TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_ITEM_TYPE);

  package PERIOD_IO is new TEXT_IO.
                           INTEGER_IO
                           (TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE);

  DEGREE_FILENAMES        : TEXT_IO.FILE_TYPE;
  DEGREE_FILENAME         : STANDARD.STRING(1..30);
  LENGTH_OF_FILENAME      : NATURAL;

  EVENT_PTR               : TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_NODE_PTR_TYPE;

  EVENT_LIST_PTR          : TIMETABLE_TYPES_PACKAGE.
                            BLACKBOARD_ITEM_PTR_TYPE;

  DEGREE_ACTIVITIES_PTR   : TIMETABLE_TYPES_PACKAGE.
                            BLACKBOARD_ITEM_PTR_TYPE;
  COMMON_REQUIREMENT_PTR  : TIMETABLE_TYPES_PACKAGE.
                            BLACKBOARD_ITEM_PTR_TYPE;
  REQUIREMENT_PTR         : TIMETABLE_TYPES_PACKAGE.
                            BLACKBOARD_ITEM_PTR_TYPE;
  DEGREE_MODULES_PTR      : TIMETABLE_TYPES_PACKAGE.
                            BLACKBOARD_ITEM_PTR_TYPE;

  EVENT_ERROR : exception;

------------------------------------------------------------------------
```

```
begin

  -- Set up the timetable blackboard

  BLACKBOARD_INITIALISE_KS_PACKAGE.
  BUILD_BLACKBOARD_LEVELS
  (TIMETABLE_BLACKBOARD_PACKAGE.BLACKBOARD);

   EVENT_LIST_PTR := TIMETABLE_BLACKBOARD_PACKAGE.
                     TIMETABLE_BLACKBOARD.
                     BLACKBOARD_ITEM
                     (BLACKBOARD  => TIMETABLE_BLACKBOARD_PACKAGE.
                                     BLACKBOARD,
                      LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                     EVENT_LISTS,
                      ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                     COMMON_EVENTS);

  TEXT_IO.OPEN
  (DEGREE_FILENAMES,
   TEXT_IO.IN_FILE,
   "plmmodfile.list");

  while not TEXT_IO.END_OF_FILE(DEGREE_FILENAMES)
  loop

    -- Get next degree name

    SYSTEM_TYPES_PACKAGE.
    GET
    (DEGREE_FILENAMES,
     DEGREE_FILENAME,
     LENGTH_OF_FILENAME);

    if not TEXT_IO.END_OF_FILE(DEGREE_FILENAMES) and then
           TEXT_IO.END_OF_LINE(DEGREE_FILENAMES) then

      TEXT_IO.SKIP_LINE(DEGREE_FILENAMES);

    end if;

    -- Add all degree modules to the blackboard

    SYLLABUS_KS_PACKAGE.
    GET_DEGREE_MODULES
    (DEGREE_FILENAME(1..LENGTH_OF_FILENAME),
     TIMETABLE_BLACKBOARD_PACKAGE.
     BLACKBOARD);

    -- Process events

   while not TIMETABLE_TYPES_PACKAGE.
             TIMETABLE_LIST_PACKAGE.
             IS_EMPTY
             (EVENT_LIST_PTR.LIST)
    loop

      TIMETABLE_TYPES_PACKAGE.
      TIMETABLE_LIST_PACKAGE.
      GET_FROM_FRONT_OF
```

```
      (LIST      => EVENT_LIST_PTR.LIST,
       THIS_ITEM => EVENT_PTR);

      case EVENT_PTR.KIND is

        when TIMETABLE_TYPES_PACKAGE.
             MODULE_REQUIREMENT_KIND =>

               COMMON_KS_PACKAGE.
               PROCESS_EVENT
               (MODULE_REQUIREMENT_PTR  => EVENT_PTR,
                BLACKBOARD              =>
                TIMETABLE_BLACKBOARD_PACKAGE.
                BLACKBOARD);

        when TIMETABLE_TYPES_PACKAGE.
             MODULE_KIND =>

               REQUIREMENT_KS_PACKAGE.
               PROCESS_EVENT
               (MODULE_PTR => EVENT_PTR,
                BLACKBOARD => TIMETABLE_BLACKBOARD_PACKAGE.
                              BLACKBOARD);

        when others => raise EVENT_ERROR;

      end case;
    end loop;
  end loop;

  EVENT_LIST_PTR := TIMETABLE_BLACKBOARD_PACKAGE.
                    TIMETABLE_BLACKBOARD.
                    BLACKBOARD_ITEM
                    (BLACKBOARD  => TIMETABLE_BLACKBOARD_PACKAGE.
                                    BLACKBOARD,
                     LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                    EVENT_LISTS,
                     ITEM_INDEX  => TIMETABLE_TYPES_PACKAGE.
                                    PERIOD_EVENTS);

   while not TIMETABLE_TYPES_PACKAGE.
             TIMETABLE_LIST_PACKAGE.
             IS_EMPTY
             (EVENT_LIST_PTR.LIST)
    loop

      TIMETABLE_TYPES_PACKAGE.
      TIMETABLE_LIST_PACKAGE.
      GET_FROM_FRONT_OF
      (LIST      => EVENT_LIST_PTR.LIST,
       THIS_ITEM => EVENT_PTR);

      case EVENT_PTR.KIND is

        when TIMETABLE_TYPES_PACKAGE.
             PERIOD_KIND =>

               null;

        when TIMETABLE_TYPES_PACKAGE.
```

```
            MODULE_ACTIVITY_KIND =>

              if SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 IS_EQUAL
                 (RIGHT => "Allocate Staff",
                  LEFT  => EVENT_PTR.
                           ACTION) then

                 STAFF_KS_PACKAGE.
                 PROCESS_EVENT
                 (ACTIVITY_PTR => EVENT_PTR,
                  BLACKBOARD   => TIMETABLE_BLACKBOARD_PACKAGE.
                                  BLACKBOARD);
              else

                 PERIOD_KS_PACKAGE.
                 PROCESS_EVENT
                 (ACTIVITY_PTR => EVENT_PTR,
                  BLACKBOARD   => TIMETABLE_BLACKBOARD_PACKAGE.
                                  BLACKBOARD);

              end if;

         when TIMETABLE_TYPES_PACKAGE.
              MODULE_COMMON_REQUIREMENT_KIND =>

              ACTIVITY_KS_PACKAGE.
              PROCESS_EVENT
              (COMMON_MODULE_PTR  => EVENT_PTR,
               BLACKBOARD         => TIMETABLE_BLACKBOARD_PACKAGE.
                                     BLACKBOARD);

         when others => raise EVENT_ERROR;

       end case;

    end loop;

    -- Form timetable files

    TIMETABLE_KS_PACKAGE.
    PROCESS_EVENT
    (BLACKBOARD => TIMETABLE_BLACKBOARD_PACKAGE.
                   BLACKBOARD);

    TEXT_IO.CLOSE(DEGREE_FILENAMES);
-------------------------------------------------------------------
exception
  when EVENT_ERROR =>
    TEXT_IO.
    PUT_LINE("Exception EVENT_ERROR in TIMETABLE_SCHEDULER");
-------------------------------------------------------------------
end TIMETABLE_SCHEDULER;
-------------------------------------------------------------------
```

Move KS

```
-------------------------------------------------------------------
--
-- Unit      : MOVE_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 30 October 1993
-- Function  : This package provides the timetable move operations.
--             These allow a module and its common modules to be
--             moved into a vacant period
--
-------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     STAFF_PACKAGE,
     MODULE_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     ROOM_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE,
     PERIOD_ESE_PACKAGE,
     PERIOD_IT_PACKAGE;
use  PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE,
     PERIOD_ESE_PACKAGE,
     PERIOD_IT_PACKAGE;
-------------------------------------------------------------------
package MOVE_PACKAGE is
-------------------------------------------------------------------

  procedure MOVE_TO
  (FM_ACTIVITY_NAME  : in     STANDARD.STRING;
   FM_DAY            : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_ITEM_TYPE;
   FM_PERIOD         : in     TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   TO_DAY            : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_ITEM_TYPE;
   TO_PERIOD         : in     TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   BLACKBOARD        : in     TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_TYPE;
   SUCCESS           :    out BOOLEAN);

-------------------------------------------------------------------
end MOVE_PACKAGE;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : MOVE_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,.
--             Cranfield University, RMCS, Shrivenham
-- Date      : 30 October 1993
-- Function  : This package provides the blackboard move operations
--
-------------------------------------------------------------------
package body MOVE_PACKAGE is
-------------------------------------------------------------------
use TIMETABLE_TYPES_PACKAGE;
  -----------------------------------------------------------------
  procedure ADD_SUPPORTER
  -----------------------------------------------------------------
  (TO_PERIOD_PTR   : in TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE;
   FM_ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE) is
  -----------------------------------------------------------------
  begin
    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST      => TO_PERIOD_PTR.
                  SUPPORTERS,
     THIS_ITEM => FM_ACTIVITY_PTR);
  -----------------------------------------------------------------
  end ADD_SUPPORTER;
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  procedure GET
  -----------------------------------------------------------------
  (FRAME_PTR    : in out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_FRAME_BASE_PACKAGE.
                         FRAME_PACKAGE.
                         FRAME_RECORD_PTR_TYPE;
   PERIOD_PTR   : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE) is
  -----------------------------------------------------------------
  begin
    TIMETABLE_TYPES_PACKAGE.
    PERIOD_FRAME_BASE_PACKAGE.
    FIND_FRAME
    (FRAME_NAME         => ACTIVITY_PTR.
                           FACT,
     FRAME_BASE_RECORD => PERIOD_PTR.
                           PERIOD_DETAIL_FRAMES,
     FRAME_PTR          => FRAME_PTR);
  -----------------------------------------------------------------
  end GET;
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  procedure DELETE_SUPPORTER
```

```
-------------------------------------------------------------------
(FM_PERIOD_PTR    : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 FM_ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
begin
  TIMETABLE_TYPES_PACKAGE.
  TIMETABLE_LIST_PACKAGE.
  DELETE_FROM
  (LIST       => FM_PERIOD_PTR.
                 SUPPORTERS,
   THIS_ITEM => FM_ACTIVITY_PTR);

-------------------------------------------------------------------
end DELETE_SUPPORTER;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(ACTIVITY_PTR  :    out TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE;
 PERIOD_PTR    : in     TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE;
 ACTIVITY_NAME : in     STANDARD.STRING) is
-------------------------------------------------------------------
  TEMP_ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE;
-------------------------------------------------------------------
begin
  for ACTIVITY_NUMBER in 1 .. TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_LIST_PACKAGE.
                              LENGTH_OF
                              (LIST => PERIOD_PTR.
                                       SUPPORTERS)
  loop
    TEMP_ACTIVITY_PTR := TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_LIST_PACKAGE.
                         ITEM_AT
                         (LIST        => PERIOD_PTR.
                                         SUPPORTERS,
                          NODE_NUMBER => ACTIVITY_NUMBER);

    if SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       SUBSTRING_OF
       (THE_STRING => TEMP_ACTIVITY_PTR.FACT) = ACTIVITY_NAME then

      -- Activity is the prime module

      ACTIVITY_PTR := TEMP_ACTIVITY_PTR;

      return;

    else

      -- Check if it is a common activity
```

```
      for COMMON_ACTIVITY_NUMBER in
          1 .. SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LENGTH_OF
               (LIST => TEMP_ACTIVITY_PTR.
                        LIST_OF_COMMON_ACTIVITIES)
      loop

        if SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           SUBSTRING_OF
           (THE_STRING => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          ITEM_AT
                          (LIST        =>
                           TEMP_ACTIVITY_PTR.
                           LIST_OF_COMMON_ACTIVITIES,
                           NODE_NUMBER =>
                           COMMON_ACTIVITY_NUMBER)) =
                          ACTIVITY_NAME then

          ACTIVITY_PTR := TEMP_ACTIVITY_PTR;
          return;

        end if;
      end loop;
    end if;
  end loop;

  ACTIVITY_PTR := null; -- No activity found with this name!
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(PERIOD_PTR  :    out TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE;
 DAY         : in     TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_ITEM_TYPE;
 PERIOD      : in     TIMETABLE_TYPES_PACKAGE.
                      PERIOD_NUMBER_TYPE;
 BLACKBOARD  : in     TIMETABLE_BLACKBOARD_PACKAGE.
                      TIMETABLE_BLACKBOARD.
                      BLACKBOARD_TYPE) is
-------------------------------------------------------------------
begin
  PERIOD_PTR := TIMETABLE_BLACKBOARD_PACKAGE.
                TIMETABLE_BLACKBOARD.
                BLACKBOARD_ITEM
                (BLACKBOARD  => BLACKBOARD,
                 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                DAYS,
                 ITEM_INDEX  => DAY).
                 PERIOD(PERIOD);
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure MOVE_ACTIVITY
-------------------------------------------------------------------
(FM_PERIOD_PTR    : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 FM_ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 FM_DAY           : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_ITEM_TYPE;
 FM_PERIOD        : in TIMETABLE_TYPES_PACKAGE.
                       PERIOD_NUMBER_TYPE;
 TO_DAY           : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_ITEM_TYPE;
 TO_PERIOD        : in TIMETABLE_TYPES_PACKAGE.
                       PERIOD_NUMBER_TYPE;
 BLACKBOARD       : in TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
                       BLACKBOARD_TYPE) is
-------------------------------------------------------------------
  FM_FRAME_PTR : TIMETABLE_TYPES_PACKAGE.
                 PERIOD_FRAME_BASE_PACKAGE.
                 FRAME_PACKAGE.
                 FRAME_RECORD_PTR_TYPE;

  TO_PERIOD_PTR : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;

  ROOT : SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         STRING;

-------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "ROOT",
   TO_THE_STRING      => ROOT);

  -- Delete supporter of loosing period

  DELETE_SUPPORTER
  (FM_PERIOD_PTR   => FM_PERIOD_PTR,
   FM_ACTIVITY_PTR => FM_ACTIVITY_PTR);

  -- Get the frame pointer for activity being moved

  GET
  (FRAME_PTR    => FM_FRAME_PTR,
   PERIOD_PTR   => FM_PERIOD_PTR,
   ACTIVITY_PTR => FM_ACTIVITY_PTR);

  -- Delete the activity frame from loosing period

  TIMETABLE_TYPES_PACKAGE.
  PERIOD_FRAME_BASE_PACKAGE.
  DELETE_FRAME
```

```
   (FRAME_PTR          => FM_FRAME_PTR,
    SUPERCLASS_NAME    => ROOT,
    FRAME_BASE_RECORD => FM_PERIOD_PTR.
                         PERIOD_DETAIL_FRAMES);

   -- Get the period pointer for receiving period

   GET
   (PERIOD_PTR => TO_PERIOD_PTR,
    DAY        => TO_DAY,
    PERIOD     => TO_PERIOD,
    BLACKBOARD => BLACKBOARD);

   -- Add activity as supporter of receiving period

   ADD_SUPPORTER
   (TO_PERIOD_PTR   => TO_PERIOD_PTR,
    FM_ACTIVITY_PTR => FM_ACTIVITY_PTR);

  -- Add activity frame to receiving period

   TIMETABLE_TYPES_PACKAGE.
   PERIOD_FRAME_BASE_PACKAGE.
   ADD_FRAME
   (FRAME_PTR          => FM_FRAME_PTR,
    SUPERCLASS_NAME    => ROOT,
    FRAME_BASE_RECORD => TO_PERIOD_PTR.
                         PERIOD_DETAIL_FRAMES);

  -- Re-allocate the rooms

  ROOM_PACKAGE.
  RE_ALLOCATE_ROOM(
  PERIOD_PTR    => TO_PERIOD_PTR,
  ACTIVITY_PTR  => FM_ACTIVITY_PTR,
  DAY           => FM_DAY,
  PERIOD        => FM_PERIOD);

-------------------------------------------------------------------
end MOVE_ACTIVITY;
-------------------------------------------------------------------


-------------------------------------------------------------------
function ACTIVITY_FITS
(ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
 DAY          : in TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_ITEM_TYPE;
 PERIOD       : in TIMETABLE_TYPES_PACKAGE.
                   PERIOD_NUMBER_TYPE) return BOOLEAN is
-------------------------------------------------------------------
  QUERY : STANDARD.STRING(1 .. 42) :=
          "period(   ,p ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

  IT_QUERY : IT_PERIOD_INFERENCE_PACKAGE.
             LOGIC_KB.
             KB_RECORD;

  ESE_QUERY : ESE_PERIOD_INFERENCE_PACKAGE.
```

```
                LOGIC_KB.
                KB_RECORD;

   VALUE_LIST : SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING_LIST_PACKAGE.
                LIST_TYPE;

   DEGREE : CHARACTER;

   WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                    WEEK_ARRAY_TYPE;

   DEGREE_ERROR : exception;

-------------------------------------------------------------------
begin

   if ACTIVITY_PTR = null then

     return FALSE;

   else

     DEGREE := SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               ITEM_OF
               (THE_STRING      => ACTIVITY_PTR.
                                   FACT,
                AT_THE_POSITION => 1);

     -- Set weeks required

     PERIOD_2_PACKAGE.
     CONVERT
     (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => ACTIVITY_PTR.
                                   FREQUENCY),
      WEEK_ARRAY => WEEKS_REQUIRED);

     -- Set the query string

     QUERY(8 .. 10) := SYSTEM_TYPES_PACKAGE.
                       CONVERT
                       (TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_ITEM_TYPE'
                        IMAGE
                        (DAY)(1..3));

     QUERY(13) := TIMETABLE_TYPES_PACKAGE.
                  PERIOD_NUMBER_TYPE'
                  IMAGE
                  (PERIOD)(2);

     declare
       QUERY_OFFSET : POSITIVE := 18;
     begin
       for WEEK in TIMETABLE_TYPES_PACKAGE.WEEK_NUMBER_TYPE
       loop
```

```
    if WEEKS_REQUIRED(WEEK) = TRUE then
      QUERY (WEEK + QUERY_OFFSET) := 'o';
    end if;
    QUERY_OFFSET := QUERY_OFFSET + 1;
  end loop;
end;

case DEGREE is
  when 'E' => -- Assert the query

              ESE_PERIOD_INFERENCE_PACKAGE.
              LOGIC_KB.
              ASSERT
              (IN_CLAUSE => QUERY,
               KB        => ESE_QUERY);

              -- Ask

              ESE_PERIOD_INFERENCE_PACKAGE.
              SOLVE.
              START
              (THIS_QUERY => ESE_QUERY,
               THIS_KB    => PERIOD_1_PACKAGE.
                             ESE_PERIOD_KB);

              -- Get the result

              ESE_PERIOD_INFERENCE_PACKAGE.
              CONTROL.
              GET_RESULT
              (OUT_LIST => VALUE_LIST);

              -- Check result

              if SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 IS_EQUAL
                 (LEFT  => "No",
                  RIGHT => SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           ITEM_AT
                           (LIST        => VALUE_LIST,
                            NODE_NUMBER => 1)) then

                -- No

                return FALSE;

              else

                -- Yes
                -- Release control

                ESE_PERIOD_INFERENCE_PACKAGE.
                CONTROL.
                NO_MORE;
                return TRUE;

              end if;
```

```
    when 'I' => -- Assert the query

                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => QUERY,
                 KB        => IT_QUERY);

                -- Ask

                IT_PERIOD_INFERENCE_PACKAGE.
                SOLVE.
                START
                (THIS_QUERY => IT_QUERY,
                 THIS_KB    => PERIOD_1_PACKAGE.
                               IT_PERIOD_KB);

                -- Get the result

                IT_PERIOD_INFERENCE_PACKAGE.
                CONTROL.
                GET_RESULT
                (OUT_LIST => VALUE_LIST);

                -- Check result

                if SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   IS_EQUAL
                   (LEFT  => "No",
                    RIGHT => SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_LIST_PACKAGE.
                             ITEM_AT
                             (LIST        => VALUE_LIST,
                              NODE_NUMBER => 1)) then

                  -- No

                  return FALSE;

                else

                  -- Yes
                  -- Release control

                  IT_PERIOD_INFERENCE_PACKAGE.
                  CONTROL.
                  NO_MORE;
                  return TRUE;

                end if;

      when others => raise DEGREE_ERROR;
    end case;

    return PERIOD_3_PACKAGE.
           COMMON_MODULES_AVAILABLE
           (PERIOD_QUERY_STRING => QUERY,
            ACTIVITY_PTR        => ACTIVITY_PTR);
```

```
  end if;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR => return FALSE;
-------------------------------------------------------------------
end ACTIVITY_FITS;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADJUST_CLAUSE
-------------------------------------------------------------------
(CLAUSE     : in out STANDARD.STRING;
 VALUE_LIST : in out SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING_LIST_PACKAGE.
                     LIST_TYPE) is
-------------------------------------------------------------------
begin

  -- List format -> A o B x C o D x E x .. K o

  for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          LENGTH_OF
                          (LIST => VALUE_LIST) / 2
                           -- two at a time
  loop

  declare

    ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;
    VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;

  begin

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => ITEM_DYNAMIC_STRING);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => VALUE_DYNAMIC_STRING);

    declare

      ITEM  : STANDARD.
              CHARACTER := SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           SUBSTRING_OF
                           (THE_STRING => ITEM_DYNAMIC_STRING)(1);

      ITEM_VALUE : STANDARD.
```

```
                    STRING
                    (1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING.
                          LENGTH_OF
                          (THE_STRING => VALUE_DYNAMIC_STRING))) :=
                    SYSTEM_TYPES_PACKAGE.
                    DYNAMIC_STRING.
                    SUBSTRING_OF
                    (THE_STRING => VALUE_DYNAMIC_STRING);
    begin

      case ITEM is

        when 'A' => CLAUSE(19) := ITEM_VALUE(1);
        when 'B' => CLAUSE(21) := ITEM_VALUE(1);
        when 'C' => CLAUSE(23) := ITEM_VALUE(1);
        when 'D' => CLAUSE(25) := ITEM_VALUE(1);
        when 'E' => CLAUSE(27) := ITEM_VALUE(1);
        when 'F' => CLAUSE(29) := ITEM_VALUE(1);
        when 'G' => CLAUSE(31) := ITEM_VALUE(1);
        when 'H' => CLAUSE(33) := ITEM_VALUE(1);
        when 'I' => CLAUSE(35) := ITEM_VALUE(1);
        when 'J' => CLAUSE(37) := ITEM_VALUE(1);
        when 'K' => CLAUSE(39) := ITEM_VALUE(1);
        when others => raise CONSTRAINT_ERROR;
      end case;

    end;
  end;
  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in CHANGE_PACKAGE at " &
     "ADJUST_CLAUSE");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in CHANGE_PACKAGE at " &
     "ADJUST_CLAUSE");
-------------------------------------------------------------------
end ADJUST_CLAUSE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET_KB_CLAUSE
-------------------------------------------------------------------
(CLAUSE : in out STANDARD.STRING;
 DAY    : in     TIMETABLE_TYPES_PACKAGE.
                 TIMETABLE_ITEM_TYPE;
 PERIOD : in     TIMETABLE_TYPES_PACKAGE.
                 PERIOD_NUMBER_TYPE;
 DEGREE : in     STANDARD.CHARACTER) is
-------------------------------------------------------------------

  IT_QUERY : PERIOD_1_PACKAGE.
             IT_PERIOD_INFERENCE_PACKAGE.
             LOGIC_KB.
             KB_RECORD;
```

```
  ESE_QUERY : PERIOD_1_PACKAGE.
              ESE_PERIOD_INFERENCE_PACKAGE.
              LOGIC_KB.
              KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;

  KB_ERROR     : exception;
  DEGREE_ERROR : exception;

-------------------------------------------------------------------
begin

  -- Set day

  CLAUSE(8 .. 10) := SYSTEM_TYPES_PACKAGE.
                     CONVERT
                     (TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_ITEM_TYPE'
                      IMAGE
                      (DAY)(1 .. 3));

  -- Set period

  CLAUSE(13) := TIMETABLE_TYPES_PACKAGE.
                PERIOD_NUMBER_TYPE'
                IMAGE
                (PERIOD)(2);

  case DEGREE is
    when 'E' => -- Assert query

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => CLAUSE,
                 KB        => ESE_QUERY);

                -- Ask

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                SOLVE.
                START
                (THIS_QUERY => ESE_QUERY,
                 THIS_KB    => PERIOD_1_PACKAGE.
                               ESE_PERIOD_KB);

                -- Get result

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                CONTROL.
                GET_RESULT
                (OUT_LIST => VALUE_LIST);
```

```
                -- Check result

                if SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   IS_EQUAL
                   (LEFT  => "No",
                    RIGHT => SYSTEM_TYPES_PACKAGE.
                             DYNAMIC_STRING_LIST_PACKAGE.
                             ITEM_AT
                             (LIST        => VALUE_LIST,
                              NODE_NUMBER => 1)) then

                  -- No

                  raise KB_ERROR;

                else

                  -- Yes Clear control

                  PERIOD_1_PACKAGE.
                  ESE_PERIOD_INFERENCE_PACKAGE.
                  CONTROL.
                  NO_MORE;

                end if;

when 'I' => -- Assert query

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => CLAUSE,
                 KB        => IT_QUERY);

                -- Ask

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                SOLVE.
                START
                (THIS_QUERY => IT_QUERY,
                 THIS_KB    => PERIOD_1_PACKAGE.
                               IT_PERIOD_KB);

                -- Get result

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                CONTROL.
                GET_RESULT
                (OUT_LIST => VALUE_LIST);

                -- Check result

                if SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   IS_EQUAL
```

```
                    (LEFT  => "No",
                     RIGHT => SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING_LIST_PACKAGE.
                              ITEM_AT
                              (LIST        => VALUE_LIST,
                               NODE_NUMBER => 1)) then

                  -- No

                  raise KB_ERROR;

                else

                  -- Yes Clear control

                  PERIOD_1_PACKAGE.
                  IT_PERIOD_INFERENCE_PACKAGE.
                  CONTROL.
                  NO_MORE;

                end if;

    when others => raise DEGREE_ERROR;

  end case;

  -- Adjust the clause with values just returned from KB

  ADJUST_CLAUSE
  (CLAUSE     => CLAUSE,
   VALUE_LIST => VALUE_LIST);

-------------------------------------------------------------------
exception

  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR in CHANGE_PACKAGE at " &
     "GET_KB_CLAUSE");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in CHANGE_PACKAGE at " &
     "GET_KB_CLAUSE");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in CHANGE_PACKAGE at " &
     "GET_KB_CLAUSE");
-------------------------------------------------------------------
end GET_KB_CLAUSE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_PERIODS_FREE
-------------------------------------------------------------------
(CLAUSE       : in out STANDARD.STRING;
 ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  CLAUSE_OFFSET  : POSITIVE := 18;
```

```
  WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;
-------------------------------------------------------------------
begin

  PERIOD_2_PACKAGE.
  CONVERT
  (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => ACTIVITY_PTR.FREQUENCY),
   WEEK_ARRAY => WEEKS_REQUIRED);

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      CLAUSE(WEEK + CLAUSE_OFFSET) := 'o';
    end if;
    CLAUSE_OFFSET := CLAUSE_OFFSET + 1;

  end loop;
-------------------------------------------------------------------
end SET_PERIODS_FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_PERIODS_OCCUPIED
-------------------------------------------------------------------
(CLAUSE       : in out STANDARD.STRING;
 ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  CLAUSE_OFFSET  : POSITIVE := 18;
  WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;
-------------------------------------------------------------------
begin

  PERIOD_2_PACKAGE.
  CONVERT
  (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => ACTIVITY_PTR.FREQUENCY),
   WEEK_ARRAY => WEEKS_REQUIRED);

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      CLAUSE(WEEK + CLAUSE_OFFSET) := 'x';
    end if;
    CLAUSE_OFFSET := CLAUSE_OFFSET + 1;

  end loop;
-------------------------------------------------------------------
```

```
end SET_PERIODS_OCCUPIED;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADJUST_KB
-------------------------------------------------------------------
(FM_ACTIVITY_PTR    : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
 FM_DAY             : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
 FM_PERIOD          : in     TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE;
 TO_DAY             : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
 TO_PERIOD          : in     TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE) is
-------------------------------------------------------------------
  FM_DAY_STRING : STANDARD.
                  STRING
                  (1 .. 3) := SYSTEM_TYPES_PACKAGE.
                              CONVERT
                              (DAY => TIMETABLE_TYPES_PACKAGE.
                                      TIMETABLE_ITEM_TYPE'
                                      IMAGE
                                      (FM_DAY)(1 .. 3));

  TO_DAY_STRING : STANDARD.
                  STRING
                  (1 .. 3) := SYSTEM_TYPES_PACKAGE.
                              CONVERT
                              (DAY => TIMETABLE_TYPES_PACKAGE.
                                      TIMETABLE_ITEM_TYPE'
                                      IMAGE
                                      (FM_DAY)(1 .. 3));

  FM_PERIOD_STRING : STANDARD.
                     STRING(1 .. 1) := TIMETABLE_TYPES_PACKAGE.
                                       PERIOD_NUMBER_TYPE'
                                       IMAGE
                                       (FM_PERIOD)(2..2);

  TO_PERIOD_STRING : STANDARD.
                     STRING(1 .. 1) := TIMETABLE_TYPES_PACKAGE.
                                       PERIOD_NUMBER_TYPE'
                                       IMAGE
                                       (TO_PERIOD)(2..2);

  FM_CLAUSE : STANDARD.
              STRING(1 .. 42) := "period("         &
                                 FM_DAY_STRING     &
                                 ",p"              &
                                 FM_PERIOD_STRING  &
                                 ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

  TO_CLAUSE : STANDARD.
              STRING(1 .. 42) := "period("         &
                                 TO_DAY_STRING     &
                                 ",p"              &
                                 TO_PERIOD_STRING  &
```

```
                                        ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

   DEGREE : CHARACTER := SYSTEM_TYPES_PACKAGE.
                         DYNAMIC_STRING.
                         ITEM_OF
                         (THE_STRING       => FM_ACTIVITY_PTR.
                                              FACT,
                          AT_THE_POSITION => 1);
-------------------------------------------------------------------
   DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

   GET_KB_CLAUSE
   (CLAUSE        => FM_CLAUSE,
    DAY           => FM_DAY,
    PERIOD        => FM_PERIOD,
    DEGREE        => DEGREE);

   GET_KB_CLAUSE
   (CLAUSE        => TO_CLAUSE,
    DAY           => TO_DAY,
    PERIOD        => TO_PERIOD,
    DEGREE        => DEGREE);

   case DEGREE is
     when 'E' => PERIOD_1_PACKAGE.
                 ESE_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 RETRACT
                 (CLAUSE => FM_CLAUSE,
                  KB     => PERIOD_1_PACKAGE.
                            ESE_PERIOD_KB);

                 PERIOD_1_PACKAGE.
                 ESE_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 RETRACT
                 (CLAUSE => TO_CLAUSE,
                  KB     => PERIOD_1_PACKAGE.
                            ESE_PERIOD_KB);

     when 'I' => PERIOD_1_PACKAGE.
                 IT_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 RETRACT
                 (CLAUSE => FM_CLAUSE,
                  KB     => PERIOD_1_PACKAGE.
                            IT_PERIOD_KB);

                 PERIOD_1_PACKAGE.
                 IT_PERIOD_INFERENCE_PACKAGE.
                 LOGIC_KB.
                 RETRACT
                 (CLAUSE => TO_CLAUSE,
                  KB        => PERIOD_1_PACKAGE.
                               IT_PERIOD_KB);

     when others => raise DEGREE_ERROR;
   end case;
```

```
    SET_PERIODS_FREE
    (CLAUSE       => FM_CLAUSE,
     ACTIVITY_PTR => FM_ACTIVITY_PTR);

    SET_PERIODS_OCCUPIED
    (CLAUSE       => TO_CLAUSE,
     ACTIVITY_PTR => FM_ACTIVITY_PTR);

    case DEGREE is
      when 'E' => PERIOD_1_PACKAGE.
                  ESE_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  ASSERT
                  (IN_CLAUSE => FM_CLAUSE,
                   KB        => PERIOD_1_PACKAGE.
                                ESE_PERIOD_KB);

                  PERIOD_1_PACKAGE.
                  ESE_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  ASSERT
                  (IN_CLAUSE => TO_CLAUSE,
                   KB        => PERIOD_1_PACKAGE.
                                ESE_PERIOD_KB);

      when 'I' => PERIOD_1_PACKAGE.
                  IT_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  ASSERT
                  (IN_CLAUSE => FM_CLAUSE,
                   KB     => PERIOD_1_PACKAGE.
                             IT_PERIOD_KB);

                  PERIOD_1_PACKAGE.
                  IT_PERIOD_INFERENCE_PACKAGE.
                  LOGIC_KB.
                  ASSERT
                  (IN_CLAUSE => TO_CLAUSE,
                   KB        => PERIOD_1_PACKAGE.
                                IT_PERIOD_KB);

      when others => raise DEGREE_ERROR;
    end case;
  ------------------------------------------------------------------
  exception

    when DEGREE_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception DEGREE_ERROR in CHANGE_PACKAGE at " &
       "GET_KB_CLAUSE");
    when CONSTRAINT_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception CONSTRAINT_ERROR in CHANGE_PACKAGE at " &
       "GET_KB_CLAUSE");
    when others =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in CHANGE_PACKAGE at " &
       "GET_KB_CLAUSE");
  ------------------------------------------------------------------
```

```
end ADJUST_KB;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADJUST_COMMON_MODULE_KB
-------------------------------------------------------------------
(FM_ACTIVITY_PTR    : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
 FM_DAY             : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
 FM_PERIOD          : in     TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE;
 TO_DAY             : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
 TO_PERIOD          : in     TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE) is
-------------------------------------------------------------------
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  for COMMON_MODULE in 1 .. SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LENGTH_OF
                            (LIST => FM_ACTIVITY_PTR.
                                     LIST_OF_COMMON_ACTIVITIES)
  loop

    case SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         ITEM_OF
         (THE_STRING       => FM_ACTIVITY_PTR.
                              FACT,
          AT_THE_POSITION => 1) is

      when 'E' => case PERIOD_3_PACKAGE.
                       DEGREE
                       (ACTIVITY_PTR => FM_ACTIVITY_PTR,
                        NODE         => COMMON_MODULE) is

                    when 'E' => null;

                    when 'I' => ADJUST_KB
                                (FM_ACTIVITY_PTR =>
                                 FM_ACTIVITY_PTR,
                                 FM_DAY          => FM_DAY,
                                 FM_PERIOD       => FM_PERIOD,
                                 TO_DAY          => TO_DAY,
                                 TO_PERIOD       => TO_PERIOD);

                    when others => raise DEGREE_ERROR;

                  end case;

      when 'I' => case PERIOD_3_PACKAGE.
                       DEGREE
                       (ACTIVITY_PTR => FM_ACTIVITY_PTR,
                        NODE         => COMMON_MODULE) is
```

```
                    when 'E' => ADJUST_KB
                                (FM_ACTIVITY_PTR =>
                                 FM_ACTIVITY_PTR,
                                 FM_DAY          => FM_DAY,
                                 FM_PERIOD       => FM_PERIOD,
                                 TO_DAY          => TO_DAY,
                                 TO_PERIOD       => TO_PERIOD);

                    when 'I' => null;

                    when others => raise DEGREE_ERROR;

                  end case;

      when others => raise DEGREE_ERROR;

    end case;
  end loop;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR in CHANGE_PACKAGE at" &
     "ADJUST_COMMON_MODULE_KB");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in CHANGE_PACKAGE at" &
     "ADJUST_COMMON_MODULE_KB");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in CHANGE_PACKAGE at " &
                    "ADJUST_COMMON_MODULE_KB");
-------------------------------------------------------------------
end ADJUST_COMMON_MODULE_KB;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure MOVE_TO
-------------------------------------------------------------------
(FM_ACTIVITY_NAME  : in     STANDARD.STRING;
 FM_DAY            : in     TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_ITEM_TYPE;
 FM_PERIOD         : in     TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
 TO_DAY            : in     TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_ITEM_TYPE;
 TO_PERIOD         : in     TIMETABLE_TYPES_PACKAGE.
                            PERIOD_NUMBER_TYPE;
 BLACKBOARD        : in     TIMETABLE_BLACKBOARD_PACKAGE.
                            TIMETABLE_BLACKBOARD.
                            BLACKBOARD_TYPE;
 SUCCESS           :    out BOOLEAN) is
-------------------------------------------------------------------
  FM_PERIOD_PTR   : TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_NODE_PTR_TYPE;
  FM_ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                    TIMETABLE_NODE_PTR_TYPE;
  WEEK_ARRAY      : TIMETABLE_TYPES_PACKAGE.
                    WEEK_ARRAY_TYPE;
```

```
-------------------------------------------------------------------
begin

  if FM_ACTIVITY_NAME'LENGTH = 0 then

     SUCCESS := FALSE;

  else

    -- Get the period ptr from which the activity will be moved

    GET
    (PERIOD_PTR  => FM_PERIOD_PTR,
     DAY         => FM_DAY,
     PERIOD      => FM_PERIOD,
     BLACKBOARD  => BLACKBOARD);

    -- Get the activity pointer of activity to be moved

    GET
    (ACTIVITY_PTR  => FM_ACTIVITY_PTR,
     PERIOD_PTR    => FM_PERIOD_PTR,
     ACTIVITY_NAME => FM_ACTIVITY_NAME);

    CONVERT
    (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   SUBSTRING_OF
                   (FM_ACTIVITY_PTR.
                    FREQUENCY),
     WEEK_ARRAY => WEEK_ARRAY);

    if ACTIVITY_FITS
       (ACTIVITY_PTR => FM_ACTIVITY_PTR,
        DAY          => TO_DAY,
        PERIOD       => TO_PERIOD) and then

      STAFF_PACKAGE.
      ALL_STAFF_FREE
      (STAFF_LIST                => FM_ACTIVITY_PTR.
                                    STAFF_LIST,
       MODULE                    => FM_ACTIVITY_PTR.
                                    FACT,
       DAY                       => CONVERT(DAY => TO_DAY),
       PERIOD                    => TO_PERIOD,
       WEEK_ARRAY                => WEEK_ARRAY,
       STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                    STAFF_FRAME_BASE_RECORD) then

      MOVE_ACTIVITY
      (FM_PERIOD_PTR   => FM_PERIOD_PTR,
       FM_ACTIVITY_PTR => FM_ACTIVITY_PTR,
       FM_DAY          => FM_DAY,
       FM_PERIOD       => FM_PERIOD,
       TO_DAY          => TO_DAY,
       TO_PERIOD       => TO_PERIOD,
       BLACKBOARD      => BLACKBOARD);

      ADJUST_KB
      (FM_ACTIVITY_PTR => FM_ACTIVITY_PTR,
```

```
          FM_DAY          => FM_DAY,
          FM_PERIOD       => FM_PERIOD,
          TO_DAY          => TO_DAY,
          TO_PERIOD       => TO_PERIOD);

        ADJUST_COMMON_MODULE_KB
        (FM_ACTIVITY_PTR => FM_ACTIVITY_PTR,
         FM_DAY          => FM_DAY,
         FM_PERIOD       => FM_PERIOD,
         TO_DAY          => TO_DAY,
         TO_PERIOD       => TO_PERIOD);

        -- Make staff free in fm_day and from period

        STAFF_PACKAGE.
        MAKE_ALL_STAFF_FREE
        (STAFF_LIST               => FM_ACTIVITY_PTR.
                                     STAFF_LIST,
         MODULE                   => FM_ACTIVITY_PTR.
                                     FACT,
         DAY                      => CONVERT(DAY => FM_DAY),
         PERIOD                   => FM_PERIOD,
         WEEK_ARRAY               => WEEK_ARRAY,
         STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                     STAFF_FRAME_BASE_RECORD);

        STAFF_PACKAGE.
        MAKE_ALL_STAFF_BUSY
        (STAFF_LIST               => FM_ACTIVITY_PTR.
                                     STAFF_LIST,
         MODULE                   => FM_ACTIVITY_PTR.
                                     FACT,
         DAY                      => CONVERT(DAY => TO_DAY),
         PERIOD                   => TO_PERIOD,
         WEEK_ARRAY               => WEEK_ARRAY,
         STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                     STAFF_FRAME_BASE_RECORD);

        SUCCESS := TRUE;

      else

        SUCCESS := FALSE;

      end if;
    end if;
  ------------------------------------------------------------------
  exception
    when others => SUCCESS := FALSE;
  ------------------------------------------------------------------
  end MOVE_TO;
  ------------------------------------------------------------------

------------------------------------------------------------------
end MOVE_PACKAGE;
------------------------------------------------------------------
```

Swap KS

```
------------------------------------------------------------------------
--
-- Unit      : SWAP_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 22 NOVEMBER 1993
-- Function : This package provides the timetable swap operations
--
------------------------------------------------------------------------
with TEXT_IO,
     SYSTEM_TYPES_PACKAGE,
     TIMETABLE_TYPES_PACKAGE,
     TIMETABLE_BLACKBOARD_PACKAGE,
     LOGIC_INFERENCE_PACKAGE,
     STAFF_PACKAGE,
     PERIOD_1_PACKAGE,
     PERIOD_2_PACKAGE,
     PERIOD_3_PACKAGE,
     PERIOD_4_PACKAGE,
     MOVE_PACKAGE;
------------------------------------------------------------------------
package SWAP_PACKAGE is
------------------------------------------------------------------------

  ----------------------------------------------------------------------
  procedure TRY_SWAPPING_PERIODS
  ----------------------------------------------------------------------
  (FM_ACTIVITY_NAME  : in     STANDARD.STRING;
   FM_DAY            : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_ITEM_TYPE;
   FM_PERIOD         : in     TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   TO_ACTIVITY_NAME  : in     STANDARD.STRING;
   TO_DAY            : in     TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_ITEM_TYPE;
   TO_PERIOD         : in     TIMETABLE_TYPES_PACKAGE.
                              PERIOD_NUMBER_TYPE;
   BLACKBOARD        : in     TIMETABLE_BLACKBOARD_PACKAGE.
                              TIMETABLE_BLACKBOARD.
                              BLACKBOARD_TYPE;
   SUCCESS           :    out BOOLEAN);

------------------------------------------------------------------------
end SWAP_PACKAGE;
------------------------------------------------------------------------
```

```
-------------------------------------------------------------------
--
-- Unit      : SWAP_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 24 November 1993
-- Function  : This package provides the timetable swapping
--             operations
--
-------------------------------------------------------------------
with ROOM_PACKAGE;
-------------------------------------------------------------------
package body SWAP_PACKAGE is
-------------------------------------------------------------------
    use TIMETABLE_TYPES_PACKAGE;
  -----------------------------------------------------------------
  procedure ADD_SUPPORTER
  -----------------------------------------------------------------
  (TO_PERIOD_PTR   : in TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE;
   FM_ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_NODE_PTR_TYPE) is
  -----------------------------------------------------------------
  begin
    TIMETABLE_TYPES_PACKAGE.
    TIMETABLE_LIST_PACKAGE.
    PUT_ON_BACK_OF
    (LIST      => TO_PERIOD_PTR.
                  SUPPORTERS,
     THIS_ITEM => FM_ACTIVITY_PTR);
  -----------------------------------------------------------------
  end ADD_SUPPORTER;
  -----------------------------------------------------------------


  -----------------------------------------------------------------
  procedure GET
  -----------------------------------------------------------------
  (FRAME_PTR    : in out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_FRAME_BASE_PACKAGE.
                         FRAME_PACKAGE.
                         FRAME_RECORD_PTR_TYPE;
   PERIOD_PTR   : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
   ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE) is
  -----------------------------------------------------------------
  begin
    TIMETABLE_TYPES_PACKAGE.
    PERIOD_FRAME_BASE_PACKAGE.
    FIND_FRAME
    (FRAME_NAME        => ACTIVITY_PTR.
                          FACT,
     FRAME_BASE_RECORD => PERIOD_PTR.
                          PERIOD_DETAIL_FRAMES,
     FRAME_PTR         => FRAME_PTR);
  -----------------------------------------------------------------
  end GET;
  -----------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure DELETE_SUPPORTER
-------------------------------------------------------------------
(FM_PERIOD_PTR   : in TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE;
 FM_ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
begin
  TIMETABLE_TYPES_PACKAGE.
  TIMETABLE_LIST_PACKAGE.
  DELETE_FROM
  (LIST      => FM_PERIOD_PTR.
                SUPPORTERS,
   THIS_ITEM => FM_ACTIVITY_PTR);
-------------------------------------------------------------------
end DELETE_SUPPORTER;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(ACTIVITY_PTR   :    out TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
 PERIOD_PTR     : in     TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_NODE_PTR_TYPE;
 ACTIVITY_NAME : in     STANDARD.STRING) is
-------------------------------------------------------------------
  TEMP_ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE;
-------------------------------------------------------------------
begin
  for ACTIVITY_NUMBER in 1 .. TIMETABLE_TYPES_PACKAGE.
                              TIMETABLE_LIST_PACKAGE.
                              LENGTH_OF
                              (LIST => PERIOD_PTR.
                                       SUPPORTERS)
 loop

    TEMP_ACTIVITY_PTR := TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_LIST_PACKAGE.
                         ITEM_AT
                         (LIST        => PERIOD_PTR.
                                         SUPPORTERS,
                          NODE_NUMBER => ACTIVITY_NUMBER);
    if SYSTEM_TYPES_PACKAGE.
       DYNAMIC_STRING.
       SUBSTRING_OF
       (THE_STRING => TEMP_ACTIVITY_PTR.FACT) = ACTIVITY_NAME then

      ACTIVITY_PTR := TEMP_ACTIVITY_PTR;
      -- Activity is the prime module
      return;

    else

      -- Check if it is a common activity
```

```
      for COMMON_ACTIVITY_NUMBER in
          1 .. SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LENGTH_OF
               (LIST => TEMP_ACTIVITY_PTR.
                        LIST_OF_COMMON_ACTIVITIES)
      loop

        if SYSTEM_TYPES_PACKAGE.
           DYNAMIC_STRING.
           SUBSTRING_OF
           (THE_STRING => SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          ITEM_AT
                          (LIST        =>
                           TEMP_ACTIVITY_PTR.
                           LIST_OF_COMMON_ACTIVITIES,
                           NODE_NUMBER => COMMON_ACTIVITY_NUMBER))
           =

           ACTIVITY_NAME then

          ACTIVITY_PTR := TEMP_ACTIVITY_PTR;
          -- Activity is a common module
          return;

        end if;
      end loop;
    end if;
  end loop;

  ACTIVITY_PTR := null; -- No activity found with this name!
-------------------------------------------------------------------
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET
-------------------------------------------------------------------
(PERIOD_PTR  :    out TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_NODE_PTR_TYPE;
 DAY         : in     TIMETABLE_TYPES_PACKAGE.
                      TIMETABLE_ITEM_TYPE;
 PERIOD      : in     TIMETABLE_TYPES_PACKAGE.
                      PERIOD_NUMBER_TYPE;
 BLACKBOARD  : in     TIMETABLE_BLACKBOARD_PACKAGE.
                      TIMETABLE_BLACKBOARD.
                      BLACKBOARD_TYPE) is
-------------------------------------------------------------------
begin
  PERIOD_PTR := TIMETABLE_BLACKBOARD_PACKAGE.
                TIMETABLE_BLACKBOARD.
                BLACKBOARD_ITEM
                (BLACKBOARD  => BLACKBOARD,
                 LEVEL_INDEX => TIMETABLE_TYPES_PACKAGE.
                                DAYS,
                   ITEM_INDEX  => DAY).
                   PERIOD(PERIOD);
-------------------------------------------------------------------
```

```
end GET;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_PERIODS_FREE
-------------------------------------------------------------------
(CLAUSE       : in out STANDARD.STRING;
 ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  CLAUSE_OFFSET  : POSITIVE := 18;
  WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;
-------------------------------------------------------------------
begin

  PERIOD_2_PACKAGE.
  CONVERT
  (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => ACTIVITY_PTR.FREQUENCY),
   WEEK_ARRAY => WEEKS_REQUIRED);

  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      CLAUSE(WEEK + CLAUSE_OFFSET) := 'o';
    end if;
    CLAUSE_OFFSET := CLAUSE_OFFSET + 1;

  end loop;
-------------------------------------------------------------------
end SET_PERIODS_FREE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_PERIODS_OCCUPIED
-------------------------------------------------------------------
(CLAUSE       : in out STANDARD.STRING;
 ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE) is
-------------------------------------------------------------------
  CLAUSE_OFFSET  : POSITIVE := 18;
  WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;
-------------------------------------------------------------------
begin

  PERIOD_2_PACKAGE.
  CONVERT
  (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => ACTIVITY_PTR.FREQUENCY),
   WEEK_ARRAY => WEEKS_REQUIRED);
```

```
  for WEEK in TIMETABLE_TYPES_PACKAGE.
              WEEK_NUMBER_TYPE
  loop

    if WEEKS_REQUIRED(WEEK) = TRUE then
      CLAUSE(WEEK + CLAUSE_OFFSET) := 'x';
    end if;
    CLAUSE_OFFSET := CLAUSE_OFFSET + 1;

  end loop;
-------------------------------------------------------------------
end SET_PERIODS_OCCUPIED;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADJUST_CLAUSE
-------------------------------------------------------------------
(CLAUSE     : in out STANDARD.STRING;
 VALUE_LIST : in out SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING_LIST_PACKAGE.
                     LIST_TYPE) is
-------------------------------------------------------------------
begin

  -- List format -> A o B x C o D x E x .. K o

  for ITEM_NUMBER in 1 .. SYSTEM_TYPES_PACKAGE.
                          DYNAMIC_STRING_LIST_PACKAGE.
                          LENGTH_OF
                          (LIST => VALUE_LIST) / 2
                          -- two at a time
  loop

  declare

    ITEM_DYNAMIC_STRING  : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;
    VALUE_DYNAMIC_STRING : SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING.
                           STRING;

  begin

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => ITEM_DYNAMIC_STRING);

    SYSTEM_TYPES_PACKAGE.
    DYNAMIC_STRING_LIST_PACKAGE.
    GET_FROM_FRONT_OF
    (LIST      => VALUE_LIST,
     THIS_ITEM => VALUE_DYNAMIC_STRING);

    declare
```

```
      ITEM   : STANDARD.
               CHARACTER := SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING.
                            SUBSTRING_OF
                            (THE_STRING => ITEM_DYNAMIC_STRING)(1);

      ITEM_VALUE : STANDARD.
                   STRING
                   (1 .. POSITIVE(SYSTEM_TYPES_PACKAGE.
                                  DYNAMIC_STRING.
                                  LENGTH_OF
                                 (THE_STRING =>
                                  VALUE_DYNAMIC_STRING))) :=
                   SYSTEM_TYPES_PACKAGE.
                   DYNAMIC_STRING.
                   SUBSTRING_OF
                   (THE_STRING => VALUE_DYNAMIC_STRING);
    begin

      case ITEM is

        when 'A' => CLAUSE(19) := ITEM_VALUE(1);
        when 'B' => CLAUSE(21) := ITEM_VALUE(1);
        when 'C' => CLAUSE(23) := ITEM_VALUE(1);
        when 'D' => CLAUSE(25) := ITEM_VALUE(1);
        when 'E' => CLAUSE(27) := ITEM_VALUE(1);
        when 'F' => CLAUSE(29) := ITEM_VALUE(1);
        when 'G' => CLAUSE(31) := ITEM_VALUE(1);
        when 'H' => CLAUSE(33) := ITEM_VALUE(1);
        when 'I' => CLAUSE(35) := ITEM_VALUE(1);
        when 'J' => CLAUSE(37) := ITEM_VALUE(1);
        when 'K' => CLAUSE(39) := ITEM_VALUE(1);
        when others => raise CONSTRAINT_ERROR;
      end case;

    end;
  end;
  end loop;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "ADJUST_CLAUSE");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE " &
     "at ADJUST_CLAUSE");
-------------------------------------------------------------------
end ADJUST_CLAUSE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure RESTORE_KB
-------------------------------------------------------------------
(CURRENT_CLAUSE : in STANDARD.STRING;
 OLD_CLAUSE     : in STANDARD.STRING;
 DEGREE         : in STANDARD.CHARACTER) is
-------------------------------------------------------------------
```

```
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  case DEGREE is
    when 'E' => PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                RETRACT
                (CLAUSE => CURRENT_CLAUSE,
                 KB     => PERIOD_1_PACKAGE.
                           ESE_PERIOD_KB);

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => OLD_CLAUSE,
                 KB        => PERIOD_1_PACKAGE.
                              ESE_PERIOD_KB);
    when 'I' => PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                RETRACT
                (CLAUSE => CURRENT_CLAUSE,
                 KB     => PERIOD_1_PACKAGE.
                           IT_PERIOD_KB);

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => OLD_CLAUSE,
                 KB        => PERIOD_1_PACKAGE.
                              IT_PERIOD_KB);
    when others => raise DEGREE_ERROR;
  end case;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR raised in SWAP_PACKAGE at " &
     "RESTORE_KB");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "RESTORE_KB");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "RESTORE_KB");
-------------------------------------------------------------------
end RESTORE_KB;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure ADJUST_KB
-------------------------------------------------------------------
(OLD_CLAUSE    : in STANDARD.STRING;
```

```
 ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_NODE_PTR_TYPE;
 DEGREE       : in STANDARD.CHARACTER) is
-------------------------------------------------------------------
  NEW_CLAUSE : STANDARD.
               STRING(1.. OLD_CLAUSE'LENGTH) := OLD_CLAUSE;

  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  case DEGREE is
    when 'E' => -- Retract old version first

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                RETRACT
                (CLAUSE => OLD_CLAUSE,
                 KB     => PERIOD_1_PACKAGE.
                           ESE_PERIOD_KB);

                -- Set periods to occupied

                SET_PERIODS_OCCUPIED
                (CLAUSE       => NEW_CLAUSE,
                 ACTIVITY_PTR => ACTIVITY_PTR);

                -- Assert adjusted version

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => NEW_CLAUSE,
                 KB        => PERIOD_1_PACKAGE.
                              ESE_PERIOD_KB);

    when 'I' => -- Retract old version first

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                RETRACT
                (CLAUSE => OLD_CLAUSE,
                 KB     => PERIOD_1_PACKAGE.
                           IT_PERIOD_KB);

                -- Set periods to occupied

                SET_PERIODS_OCCUPIED
                (CLAUSE       => NEW_CLAUSE,
                 ACTIVITY_PTR => ACTIVITY_PTR);

                -- Assert adjusted version

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
```

```
                (IN_CLAUSE => NEW_CLAUSE,
                 KB        => PERIOD_1_PACKAGE.
                              IT_PERIOD_KB);
    when others => raise DEGREE_ERROR;
  end case;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR raised in SWAP_PACKAGE at " &
     "ADJUST_KB");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "ADJUST_KB");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "ADJUST_KB");
-------------------------------------------------------------------
end ADJUST_KB;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure CHANGE_KB_CLAUSE
-------------------------------------------------------------------
(OLD_CLAUSE    : in     STANDARD.STRING;
 NEW_CLAUSE    : in out STANDARD.STRING;
 ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 DEGREE        : in     STANDARD.CHARACTER) is
-------------------------------------------------------------------
  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  NEW_CLAUSE := OLD_CLAUSE;

  -- Adjust new clause by setting those weeks required
  -- by activity back to 'o' unoccupied

  SET_PERIODS_FREE
  (CLAUSE       => NEW_CLAUSE,
   ACTIVITY_PTR => ACTIVITY_PTR);

  case DEGREE is
    when 'E' => PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                RETRACT
                (CLAUSE => OLD_CLAUSE,
                 KB     => PERIOD_1_PACKAGE.
                           ESE_PERIOD_KB);

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => NEW_CLAUSE,
```

```
                KB          => PERIOD_1_PACKAGE.
                               ESE_PERIOD_KB);

    when 'I' => PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                RETRACT
                (CLAUSE => OLD_CLAUSE,
                 KB     => PERIOD_1_PACKAGE.
                           IT_PERIOD_KB);

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => NEW_CLAUSE,
                 KB        => PERIOD_1_PACKAGE.
                              IT_PERIOD_KB);
    when others => raise DEGREE_ERROR;
  end case;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR raised in SWAP_PACKAGE at " &
     "CHANGE_KB_CLAUSE");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "CHANGE_KB_CLAUSE");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "CHANGE_KB_CLAUSE");
-------------------------------------------------------------------
end CHANGE_KB_CLAUSE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure  SWAP_PERIODS
-------------------------------------------------------------------
(FM_PERIOD_PTR    : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 FM_ACTIVITY_PTR  : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 FM_DAY           : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_ITEM_TYPE;
 FM_PERIOD        : in TIMETABLE_TYPES_PACKAGE.
                       PERIOD_NUMBER_TYPE;
 TO_PERIOD_PTR    : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 TO_ACTIVITY_PTR  : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 TO_DAY           : in TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_ITEM_TYPE;
 TO_PERIOD        : in TIMETABLE_TYPES_PACKAGE.
                       PERIOD_NUMBER_TYPE;
 BLACKBOARD       : in TIMETABLE_BLACKBOARD_PACKAGE.
                       TIMETABLE_BLACKBOARD.
```

```
                        BLACKBOARD_TYPE) is
-----------------------------------------------------------------------
  ROOT : SYSTEM_TYPES_PACKAGE.
         DYNAMIC_STRING.
         STRING;

  FM_FRAME_PTR : TIMETABLE_TYPES_PACKAGE.
                 PERIOD_FRAME_BASE_PACKAGE.
                 FRAME_PACKAGE.
                 FRAME_RECORD_PTR_TYPE;

  TO_FRAME_PTR : TIMETABLE_TYPES_PACKAGE.
                 PERIOD_FRAME_BASE_PACKAGE.
                 FRAME_PACKAGE.
                 FRAME_RECORD_PTR_TYPE;
-----------------------------------------------------------------------
begin

  SYSTEM_TYPES_PACKAGE.
  DYNAMIC_STRING.
  COPY
  (FROM_THE_SUBSTRING => "ROOT",
   TO_THE_STRING      => ROOT);

  -- Delete supportere of 'fm_period'

  DELETE_SUPPORTER
  (FM_PERIOD_PTR   => FM_PERIOD_PTR,
   FM_ACTIVITY_PTR => FM_ACTIVITY_PTR);

  -- Delete supporter of 'to_period'

  DELETE_SUPPORTER
  (FM_PERIOD_PTR   => TO_PERIOD_PTR,
   FM_ACTIVITY_PTR => TO_ACTIVITY_PTR);

  -- Get frame pointer of 'from' module

  GET
  (FRAME_PTR     => FM_FRAME_PTR,
   PERIOD_PTR    => FM_PERIOD_PTR,
   ACTIVITY_PTR  => FM_ACTIVITY_PTR);

  -- Get frame ptr of 'to' module

  GET
  (FRAME_PTR     => TO_FRAME_PTR,
   PERIOD_PTR    => TO_PERIOD_PTR,
   ACTIVITY_PTR  => TO_ACTIVITY_PTR);

  -- Delete activity in 'from' module frame base

  TIMETABLE_TYPES_PACKAGE.
  PERIOD_FRAME_BASE_PACKAGE.
  DELETE_FRAME
  (FRAME_PTR          => FM_FRAME_PTR,
   SUPERCLASS_NAME    => ROOT,
   FRAME_BASE_RECORD  => FM_PERIOD_PTR.
                         PERIOD_DETAIL_FRAMES);
```

```
   -- Delete activity in 'to' module frame base

   TIMETABLE_TYPES_PACKAGE.
   PERIOD_FRAME_BASE_PACKAGE.
   DELETE_FRAME
   (FRAME_PTR          => TO_FRAME_PTR,
    SUPERCLASS_NAME    => ROOT,
    FRAME_BASE_RECORD => TO_PERIOD_PTR.
                         PERIOD_DETAIL_FRAMES);

   -- Add 'fm' activity to 'to_period'

   ADD_SUPPORTER
   (TO_PERIOD_PTR    => TO_PERIOD_PTR,
    FM_ACTIVITY_PTR => FM_ACTIVITY_PTR);

   -- Add 'to' activity to 'fm_period'

   ADD_SUPPORTER
   (TO_PERIOD_PTR    => FM_PERIOD_PTR,
    FM_ACTIVITY_PTR => TO_ACTIVITY_PTR);

   -- Add 'fm' frame to 'to' frame base

   TIMETABLE_TYPES_PACKAGE.
   PERIOD_FRAME_BASE_PACKAGE.
   ADD_FRAME
   (FRAME_PTR          => FM_FRAME_PTR,
    SUPERCLASS_NAME    => ROOT,
    FRAME_BASE_RECORD => TO_PERIOD_PTR.
                         PERIOD_DETAIL_FRAMES);

   -- Add 'to' frame to 'fm' frame base

   TIMETABLE_TYPES_PACKAGE.
   PERIOD_FRAME_BASE_PACKAGE.
   ADD_FRAME
   (FRAME_PTR          => TO_FRAME_PTR,
    SUPERCLASS_NAME    => ROOT,
    FRAME_BASE_RECORD => FM_PERIOD_PTR.
                         PERIOD_DETAIL_FRAMES);

  -- Re-allocate the rooms

  ROOM_PACKAGE.
  RE_ALLOCATE_ROOM(
  PERIOD_PTR     => TO_PERIOD_PTR,
  ACTIVITY_PTR   => FM_ACTIVITY_PTR,
  DAY            => FM_DAY,
  PERIOD         => FM_PERIOD);

  ROOM_PACKAGE.
  RE_ALLOCATE_ROOM(
  PERIOD_PTR     => FM_PERIOD_PTR,
  ACTIVITY_PTR   => TO_ACTIVITY_PTR,
  DAY            => TO_DAY,
  PERIOD         => TO_PERIOD);
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
```

```
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "SWAP_PERIODS");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "SWAP_PERIODS");
-------------------------------------------------------------------
end SWAP_PERIODS;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure GET_KB_CLAUSE
-------------------------------------------------------------------
(CLAUSE       : in out STANDARD.STRING;
 ACTIVITY_PTR : in     TIMETABLE_TYPES_PACKAGE.
                       TIMETABLE_NODE_PTR_TYPE;
 DEGREE       : in     STANDARD.CHARACTER) is
-------------------------------------------------------------------
  IT_QUERY : PERIOD_1_PACKAGE.
             IT_PERIOD_INFERENCE_PACKAGE.
             LOGIC_KB.
             KB_RECORD;

  ESE_QUERY : PERIOD_1_PACKAGE.
              ESE_PERIOD_INFERENCE_PACKAGE.
              LOGIC_KB.
              KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;

  WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;

  DEGREE_ERROR : exception;
  KB_ERROR     : exception;
-------------------------------------------------------------------
begin

  -- Set weeks_required

  PERIOD_2_PACKAGE.
  CONVERT
  (WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 SUBSTRING_OF
                 (THE_STRING => ACTIVITY_PTR.
                                FREQUENCY),
   WEEK_ARRAY => WEEKS_REQUIRED);

  case DEGREE is
    when 'E'    =>  -- Assert the query

                    PERIOD_1_PACKAGE.
                    ESE_PERIOD_INFERENCE_PACKAGE.
                    LOGIC_KB.
                    ASSERT
```

```
                    (IN_CLAUSE => CLAUSE,
                     KB        => ESE_QUERY);

                  -- Ask

                   PERIOD_1_PACKAGE.
                   ESE_PERIOD_INFERENCE_PACKAGE.
                   SOLVE.
                   START
                   (THIS_QUERY => ESE_QUERY,
                    THIS_KB    => PERIOD_1_PACKAGE.
                                  ESE_PERIOD_KB);

                  -- Get the result

                   PERIOD_1_PACKAGE.
                   ESE_PERIOD_INFERENCE_PACKAGE.
                   CONTROL.
                   GET_RESULT
                   (OUT_LIST => VALUE_LIST);

                  -- Check result

                  if SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING.
                     IS_EQUAL
                     (LEFT  => "No",
                      RIGHT => SYSTEM_TYPES_PACKAGE.
                               DYNAMIC_STRING_LIST_PACKAGE.
                               ITEM_AT
                               (LIST        => VALUE_LIST,
                                NODE_NUMBER => 1)) then

                    -- No

                    raise KB_ERROR;

                  else

                    -- Yes release control

                    PERIOD_1_PACKAGE.
                    ESE_PERIOD_INFERENCE_PACKAGE.
                    CONTROL.
                    NO_MORE;

                    -- Adjust the output clause with values
                    -- just obtained

                    ADJUST_CLAUSE
                    (CLAUSE     => CLAUSE,
                     VALUE_LIST => VALUE_LIST);

                  end if;

when 'I'     =>  -- Assert the query

                   PERIOD_1_PACKAGE.
                   IT_PERIOD_INFERENCE_PACKAGE.
                   LOGIC_KB.
```

```
                    ASSERT
                    (IN_CLAUSE => CLAUSE,
                     KB        => IT_QUERY);

                   -- Ask

                    PERIOD_1_PACKAGE.
                    IT_PERIOD_INFERENCE_PACKAGE.
                    SOLVE.
                    START
                    (THIS_QUERY => IT_QUERY,
                     THIS_KB    => PERIOD_1_PACKAGE.
                                   IT_PERIOD_KB);

                   -- Get the result

                    PERIOD_1_PACKAGE.
                    IT_PERIOD_INFERENCE_PACKAGE.
                    CONTROL.
                    GET_RESULT
                    (OUT_LIST => VALUE_LIST);

                   -- Check result

                   if SYSTEM_TYPES_PACKAGE.
                      DYNAMIC_STRING.
                      IS_EQUAL
                      (LEFT  => "No",
                       RIGHT => SYSTEM_TYPES_PACKAGE.
                                DYNAMIC_STRING_LIST_PACKAGE.
                                ITEM_AT
                                (LIST        => VALUE_LIST,
                                 NODE_NUMBER => 1)) then

                     -- No

                     raise KB_ERROR;

                   else

                     -- Yes release control

                     PERIOD_1_PACKAGE.
                     IT_PERIOD_INFERENCE_PACKAGE.
                     CONTROL.
                     NO_MORE;

                     -- Adjust the output clause with values
                     -- just obtained

                     ADJUST_CLAUSE
                     (CLAUSE => CLAUSE,
                      VALUE_LIST => VALUE_LIST);

                   end if;

    when others => raise DEGREE_ERROR;

  end case;
```

```
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR raised in SWAP_PACKAGE at " &
     "GET_KB_CLAUSE");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "GET_KB_CLAUSE");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "GET_KB_CLAUSE");
-------------------------------------------------------------------
end GET_KB_CLAUSE;
-------------------------------------------------------------------


-------------------------------------------------------------------
function ACTIVITY_FITS
-------------------------------------------------------------------
(ACTIVITY_PTR : in TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;
 DAY          : in TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_ITEM_TYPE;
 PERIOD       : in TIMETABLE_TYPES_PACKAGE.
                  PERIOD_NUMBER_TYPE;
 DEGREE       : in STANDARD.CHARACTER) return BOOLEAN is
-------------------------------------------------------------------
  QUERY : STANDARD.STRING(1 .. 42) :=
          "period(   ,p ,wks(_,_,_,_,_,_,_,_,_,_,_,_)).";

  IT_QUERY : PERIOD_1_PACKAGE.
             IT_PERIOD_INFERENCE_PACKAGE.
             LOGIC_KB.
             KB_RECORD;

  ESE_QUERY : PERIOD_1_PACKAGE.
              ESE_PERIOD_INFERENCE_PACKAGE.
              LOGIC_KB.
              KB_RECORD;

  VALUE_LIST : SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING_LIST_PACKAGE.
               LIST_TYPE;

  WEEKS_REQUIRED : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;

  DEGREE_ERROR : exception;
-------------------------------------------------------------------
begin

  if ACTIVITY_PTR = null then

    return FALSE;

  else

    -- Set weeks required
```

```
PERIOD_2_PACKAGE.
CONVERT
(WEEK_CODE  => SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               SUBSTRING_OF
               (THE_STRING => ACTIVITY_PTR.
                              FREQUENCY),
 WEEK_ARRAY => WEEKS_REQUIRED);

-- Set the query string

QUERY(8 .. 10) := SYSTEM_TYPES_PACKAGE.
                  CONVERT
                  (TIMETABLE_TYPES_PACKAGE.
                   TIMETABLE_ITEM_TYPE'
                   IMAGE
                   (DAY)(1..3));

QUERY(13) := TIMETABLE_TYPES_PACKAGE.
             PERIOD_NUMBER_TYPE'
             IMAGE
             (PERIOD)(2);

declare
  QUERY_OFFSET : POSITIVE := 18;
begin
  for WEEK in TIMETABLE_TYPES_PACKAGE.WEEK_NUMBER_TYPE
  loop
    if WEEKS_REQUIRED(WEEK) = TRUE then
      QUERY (WEEK + QUERY_OFFSET) := 'o';
    end if;
    QUERY_OFFSET := QUERY_OFFSET + 1;
  end loop;
end;

 case DEGREE is
   when 'E' => -- Assert the query

               PERIOD_1_PACKAGE.
               ESE_PERIOD_INFERENCE_PACKAGE.
               LOGIC_KB.
               ASSERT
               (IN_CLAUSE => QUERY,
                KB        => ESE_QUERY);

               -- Ask

               PERIOD_1_PACKAGE.
               ESE_PERIOD_INFERENCE_PACKAGE.
               SOLVE.
               START
               (THIS_QUERY => ESE_QUERY,
                THIS_KB    => PERIOD_1_PACKAGE.
                              ESE_PERIOD_KB);

              -- Get the result

               PERIOD_1_PACKAGE.
               ESE_PERIOD_INFERENCE_PACKAGE.
```

```
                CONTROL.
                GET_RESULT
                (OUT_LIST => VALUE_LIST);

              -- Check result

              if SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 IS_EQUAL
                 (LEFT  => "No",
                  RIGHT => SYSTEM_TYPES_PACKAGE.
                           DYNAMIC_STRING_LIST_PACKAGE.
                           ITEM_AT
                           (LIST        => VALUE_LIST,
                            NODE_NUMBER => 1)) then

                -- No

                return FALSE;

              else

                -- Yes
                -- Release control

                PERIOD_1_PACKAGE.
                ESE_PERIOD_INFERENCE_PACKAGE.
                CONTROL.
                NO_MORE;
                return TRUE;

              end if;

when 'I' => -- Assert the query

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                LOGIC_KB.
                ASSERT
                (IN_CLAUSE => QUERY,
                 KB        => IT_QUERY);

                -- Ask

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                SOLVE.
                START
                (THIS_QUERY => IT_QUERY,
                 THIS_KB    => PERIOD_1_PACKAGE.
                               IT_PERIOD_KB);

              -- Get the result

                PERIOD_1_PACKAGE.
                IT_PERIOD_INFERENCE_PACKAGE.
                CONTROL.
                GET_RESULT
                (OUT_LIST => VALUE_LIST);
```

```
                    -- Check result

                    if SYSTEM_TYPES_PACKAGE.
                       DYNAMIC_STRING.
                       IS_EQUAL
                       (LEFT  => "No",
                        RIGHT => SYSTEM_TYPES_PACKAGE.
                                 DYNAMIC_STRING_LIST_PACKAGE.
                                 ITEM_AT
                                 (LIST        => VALUE_LIST,
                                  NODE_NUMBER => 1)) then

                      -- No

                      return FALSE;

                    else

                      -- Yes
                      -- Release control

                      PERIOD_1_PACKAGE.
                      IT_PERIOD_INFERENCE_PACKAGE.
                      CONTROL.
                      NO_MORE;

                      return TRUE;

                    end if;

          when others => raise DEGREE_ERROR;

        end case;

    end if;
-------------------------------------------------------------------
exception
  when DEGREE_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception DEGREE_ERROR in SWAP_PACKAGE at " &
     "ACTIVITY_FITS");
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "ACTIVITY_FITS");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "ACTIVITY_FITS");
-------------------------------------------------------------------
end ACTIVITY_FITS;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure TRY_SWAPPING_COMMON_MODULES
-------------------------------------------------------------------
(COMMON_LIST       : in     SYSTEM_TYPES_PACKAGE.
                            DYNAMIC_STRING_LIST_PACKAGE.
                            LIST_TYPE;
```

```
FM_ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
FM_PERIOD_PTR      : in      TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
FM_DAY             : in      TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
FM_PERIOD          : in      TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE;
TO_ACTIVITY_PTR    : in      TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
TO_PERIOD_PTR      : in      TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_NODE_PTR_TYPE;
TO_DAY             : in      TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
TO_PERIOD          : in      TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE;
BLACKBOARD         : in      TIMETABLE_BLACKBOARD_PACKAGE.
                             TIMETABLE_BLACKBOARD.
                             BLACKBOARD_TYPE;
SUCCESS            :    out BOOLEAN) is
-------------------------------------------------------------------
 LOCAL_SUCCESS : BOOLEAN := FALSE;

 LOCAL_COMMON_LIST : SYSTEM_TYPES_PACKAGE.
                     DYNAMIC_STRING_LIST_PACKAGE.
                     LIST_TYPE := COMMON_LIST;

 FM_DAY_STRING : STANDARD.
                 STRING(1 .. 3) :=
                 SYSTEM_TYPES_PACKAGE.
                 CONVERT
                 (DAY => TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_ITEM_TYPE'
                         IMAGE
                         (FM_DAY)(1 .. 3));

 TO_DAY_STRING : STANDARD.
                 STRING(1 .. 3) :=
                 SYSTEM_TYPES_PACKAGE.
                 CONVERT
                 (DAY => TIMETABLE_TYPES_PACKAGE.
                         TIMETABLE_ITEM_TYPE'
                         IMAGE
                         (TO_DAY)(1 .. 3));

 FM_PERIOD_STRING : STANDARD.
                    STRING(1 .. 1) := TIMETABLE_TYPES_PACKAGE.
                                      PERIOD_NUMBER_TYPE'
                                      IMAGE
                                      (FM_PERIOD)(2..2);

 TO_PERIOD_STRING : STANDARD.
                    STRING(1 .. 1) := TIMETABLE_TYPES_PACKAGE.
                                      PERIOD_NUMBER_TYPE'
                                      IMAGE
                                      (TO_PERIOD)(2..2);

 FM_CLAUSE : constant STANDARD.
                      STRING(1 .. 42) :=
                      "period("        &
```

```
                        FM_DAY_STRING     &
                        ",p"              &
                        FM_PERIOD_STRING  &
                        ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

   TO_CLAUSE : constant STANDARD.
                        STRING(1 .. 42) :=
                        "period("         &
                        TO_DAY_STRING     &
                        ",p"              &
                        TO_PERIOD_STRING  &
                        ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

   FM_OLD_CLAUSE  : STANDARD.STRING(1 ..42) := FM_CLAUSE;
   TO_OLD_CLAUSE  : STANDARD.STRING(1 ..42) := TO_CLAUSE;
   FM_TEMP_CLAUSE : STANDARD.STRING(1 ..42) := FM_CLAUSE;
   TO_TEMP_CLAUSE : STANDARD.STRING(1 ..42) := TO_CLAUSE;

   MODULE_NAME : SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 STRING;

   DEGREE : STANDARD.CHARACTER;
-------------------------------------------------------------------
begin

   if not SYSTEM_TYPES_PACKAGE.
          DYNAMIC_STRING_LIST_PACKAGE.
          IS_EMPTY
          (LIST => LOCAL_COMMON_LIST) then

     SYSTEM_TYPES_PACKAGE.
     DYNAMIC_STRING_LIST_PACKAGE.
     GET_FROM_FRONT_OF
     (LIST => LOCAL_COMMON_LIST,
      THIS_ITEM => MODULE_NAME);

     DEGREE := SYSTEM_TYPES_PACKAGE.
               DYNAMIC_STRING.
               ITEM_OF
               (THE_STRING      => MODULE_NAME,
                AT_THE_POSITION => 1);

     GET_KB_CLAUSE
     (CLAUSE       => FM_OLD_CLAUSE,
      ACTIVITY_PTR => FM_ACTIVITY_PTR,
      DEGREE       => DEGREE);

     GET_KB_CLAUSE
     (CLAUSE       => TO_OLD_CLAUSE,
      ACTIVITY_PTR => TO_ACTIVITY_PTR,
      DEGREE       => DEGREE);

     CHANGE_KB_CLAUSE
     (OLD_CLAUSE   => FM_OLD_CLAUSE,
      NEW_CLAUSE   => FM_TEMP_CLAUSE,
      ACTIVITY_PTR => FM_ACTIVITY_PTR,
      DEGREE       => DEGREE);

     CHANGE_KB_CLAUSE
```

```
(OLD_CLAUSE    => TO_OLD_CLAUSE,
 NEW_CLAUSE    => TO_TEMP_CLAUSE,
 ACTIVITY_PTR => TO_ACTIVITY_PTR,
 DEGREE        => DEGREE);

if  ACTIVITY_FITS
   (ACTIVITY_PTR => FM_ACTIVITY_PTR,
    DAY          => TO_DAY,
    PERIOD       => TO_PERIOD,
    DEGREE       => DEGREE)

    and then

    ACTIVITY_FITS
    (ACTIVITY_PTR => TO_ACTIVITY_PTR,
     DAY          => FM_DAY,
     PERIOD       => FM_PERIOD,
     DEGREE       => DEGREE) then

   TRY_SWAPPING_COMMON_MODULES
   (COMMON_LIST     => LOCAL_COMMON_LIST,
    FM_ACTIVITY_PTR => FM_ACTIVITY_PTR,
    FM_PERIOD_PTR   => FM_PERIOD_PTR,
    FM_DAY          => FM_DAY,
    FM_PERIOD       => FM_PERIOD,
    TO_ACTIVITY_PTR => TO_ACTIVITY_PTR,
    TO_PERIOD_PTR   => TO_PERIOD_PTR,
    TO_DAY          => TO_DAY,
    TO_PERIOD       => TO_PERIOD,
    BLACKBOARD      => BLACKBOARD,
    SUCCESS         => LOCAL_SUCCESS);

   if LOCAL_SUCCESS then

     ADJUST_KB
     (OLD_CLAUSE    => FM_TEMP_CLAUSE,
      ACTIVITY_PTR => FM_ACTIVITY_PTR,
      DEGREE        => DEGREE);

     ADJUST_KB
     (OLD_CLAUSE    => TO_TEMP_CLAUSE,
      ACTIVITY_PTR => TO_ACTIVITY_PTR,
      DEGREE        => DEGREE);

     SUCCESS := TRUE;

   else

     RESTORE_KB
     (CURRENT_CLAUSE => FM_TEMP_CLAUSE,
      OLD_CLAUSE     => FM_OLD_CLAUSE,
      DEGREE         => DEGREE);

     RESTORE_KB
     (CURRENT_CLAUSE => TO_TEMP_CLAUSE,
      OLD_CLAUSE     => TO_OLD_CLAUSE,
      DEGREE         => DEGREE);

     SUCCESS := FALSE;
```

```
      end if;

    else

      RESTORE_KB
      (CURRENT_CLAUSE => FM_TEMP_CLAUSE,
       OLD_CLAUSE     => FM_OLD_CLAUSE,
       DEGREE         => DEGREE);

      RESTORE_KB
      (CURRENT_CLAUSE => TO_TEMP_CLAUSE,
       OLD_CLAUSE     => TO_OLD_CLAUSE,
       DEGREE         => DEGREE);

      SUCCESS := FALSE;

    end if;

    -- No modules to check therefore ok

  else

    SUCCESS := TRUE;

  end if;
-------------------------------------------------------------------
exception
  when CONSTRAINT_ERROR =>
    TEXT_IO.PUT_LINE
    ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
     "TRY_SWAPPING_COMMON_MODULES");
  when OTHERS =>
    TEXT_IO.PUT_LINE
    ("Exception OTHERS in SWAP_PACKAGE at " &
     "TRY_SWAPPING_COMMON_MODULES");
-------------------------------------------------------------------
end TRY_SWAPPING_COMMON_MODULES;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure TRY_SWAPPING_PERIODS
-------------------------------------------------------------------
(FM_ACTIVITY_NAME   : in     STANDARD.STRING;
 FM_DAY             : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
 FM_PERIOD          : in     TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE;
 TO_ACTIVITY_NAME   : in     STANDARD.STRING;
 TO_DAY             : in     TIMETABLE_TYPES_PACKAGE.
                             TIMETABLE_ITEM_TYPE;
 TO_PERIOD          : in     TIMETABLE_TYPES_PACKAGE.
                             PERIOD_NUMBER_TYPE;
 BLACKBOARD         : in     TIMETABLE_BLACKBOARD_PACKAGE.
                             TIMETABLE_BLACKBOARD.
                             BLACKBOARD_TYPE;
 SUCCESS            :    out BOOLEAN) is
-------------------------------------------------------------------
  FM_DAY_STRING : STANDARD.
                  STRING(1 .. 3) :=
```

```
                    SYSTEM_TYPES_PACKAGE.
                    CONVERT
                    (DAY => TIMETABLE_TYPES_PACKAGE.
                            TIMETABLE_ITEM_TYPE'
                            IMAGE
                            (FM_DAY)(1 .. 3));

TO_DAY_STRING : STANDARD.
                STRING(1 .. 3) :=
                SYSTEM_TYPES_PACKAGE.
                CONVERT
                (DAY => TIMETABLE_TYPES_PACKAGE.
                        TIMETABLE_ITEM_TYPE'
                        IMAGE
                        (TO_DAY)(1 .. 3));

FM_PERIOD_STRING : STANDARD.
                   STRING(1 .. 1) := TIMETABLE_TYPES_PACKAGE.
                                     PERIOD_NUMBER_TYPE'
                                     IMAGE
                                     (FM_PERIOD)(2..2);

TO_PERIOD_STRING : STANDARD.
                   STRING(1 .. 1) := TIMETABLE_TYPES_PACKAGE.
                                     PERIOD_NUMBER_TYPE'
                                     IMAGE
                                     (TO_PERIOD)(2..2);

FM_CLAUSE : constant STANDARD.
                     STRING(1 .. 42) :=
                     "period("          &
                     FM_DAY_STRING      &
                     ",p"               &
                     FM_PERIOD_STRING &
                     ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

TO_CLAUSE : constant STANDARD.
                     STRING(1 .. 42) :=
                     "period("          &
                     TO_DAY_STRING      &
                     ",p"               &
                     TO_PERIOD_STRING &
                     ",wks(A,B,C,D,E,F,G,H,I,J,K)).";

DEGREE : CHARACTER;

FM_OLD_CLAUSE  : STANDARD.STRING(1 ..42) := FM_CLAUSE;
TO_OLD_CLAUSE  : STANDARD.STRING(1 ..42) := TO_CLAUSE;
FM_TEMP_CLAUSE : STANDARD.STRING(1 ..42) := FM_CLAUSE;
TO_TEMP_CLAUSE : STANDARD.STRING(1 ..42) := TO_CLAUSE;

FM_PERIOD_PTR   : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;
FM_ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;
TO_PERIOD_PTR   : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;
TO_ACTIVITY_PTR : TIMETABLE_TYPES_PACKAGE.
                  TIMETABLE_NODE_PTR_TYPE;
```

```
   LOCAL_SUCCESS : BOOLEAN := FALSE;

   FM_WEEK_ARRAY : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;

   TO_WEEK_ARRAY : TIMETABLE_TYPES_PACKAGE.
                   WEEK_ARRAY_TYPE;
-------------------------------------------------------------------
begin

   if FM_ACTIVITY_NAME'LENGTH = 0 or else
      TO_ACTIVITY_NAME'LENGTH = 0 then

     SUCCESS := FALSE;

   else

     GET
     (PERIOD_PTR => FM_PERIOD_PTR,
      DAY        => FM_DAY,
      PERIOD     => FM_PERIOD,
      BLACKBOARD => BLACKBOARD);

     GET
     (PERIOD_PTR => TO_PERIOD_PTR,
      DAY        => TO_DAY,
      PERIOD     => TO_PERIOD,
      BLACKBOARD => BLACKBOARD);

     GET
     (ACTIVITY_PTR  => FM_ACTIVITY_PTR,
      PERIOD_PTR    => FM_PERIOD_PTR,
      ACTIVITY_NAME => FM_ACTIVITY_NAME);

     GET
     (ACTIVITY_PTR  => TO_ACTIVITY_PTR,
      PERIOD_PTR    => TO_PERIOD_PTR,
      ACTIVITY_NAME => TO_ACTIVITY_NAME);

     if FM_ACTIVITY_PTR = null or else TO_ACTIVITY_PTR = null then

       SUCCESS := FALSE;

     else

       DEGREE := SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 ITEM_OF
                 (THE_STRING       => FM_ACTIVITY_PTR.
                                      FACT,
                  AT_THE_POSITION => 1);

       GET_KB_CLAUSE
       (CLAUSE       => FM_OLD_CLAUSE,
        ACTIVITY_PTR => FM_ACTIVITY_PTR,
        DEGREE       => DEGREE);

       GET_KB_CLAUSE
       (CLAUSE       => TO_OLD_CLAUSE,
        ACTIVITY_PTR => TO_ACTIVITY_PTR,
```

```
 DEGREE        => DEGREE);

CHANGE_KB_CLAUSE
(OLD_CLAUSE    => FM_OLD_CLAUSE,
 NEW_CLAUSE    => FM_TEMP_CLAUSE,
 ACTIVITY_PTR => FM_ACTIVITY_PTR,
 DEGREE        => DEGREE);

CHANGE_KB_CLAUSE
(OLD_CLAUSE    => TO_OLD_CLAUSE,
 NEW_CLAUSE    => TO_TEMP_CLAUSE,
 ACTIVITY_PTR => TO_ACTIVITY_PTR,
 DEGREE        => DEGREE);

PERIOD_2_PACKAGE.
CONVERT
(WEEK_CODE   => SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                SUBSTRING_OF
                (FM_ACTIVITY_PTR.
                 FREQUENCY),
 WEEK_ARRAY => FM_WEEK_ARRAY);

PERIOD_2_PACKAGE.
CONVERT
(WEEK_CODE   => SYSTEM_TYPES_PACKAGE.
                DYNAMIC_STRING.
                SUBSTRING_OF
                (TO_ACTIVITY_PTR.
                 FREQUENCY),
 WEEK_ARRAY => TO_WEEK_ARRAY);

if ACTIVITY_FITS
   (ACTIVITY_PTR => FM_ACTIVITY_PTR,
   DAY           => TO_DAY,
   PERIOD        => TO_PERIOD,
   DEGREE        => DEGREE)

   and then

   ACTIVITY_FITS
   (ACTIVITY_PTR => TO_ACTIVITY_PTR,
   DAY           => FM_DAY,
   PERIOD        => FM_PERIOD,
   DEGREE        => DEGREE) and then

   STAFF_PACKAGE.
   ALL_STAFF_FREE
   (STAFF_LIST                => FM_ACTIVITY_PTR.
                                 STAFF_LIST,
    MODULE                    => FM_ACTIVITY_PTR.
                                 FACT,
    DAY                       => PERIOD_3_PACKAGE.
                                 CONVERT(DAY => TO_DAY),
    PERIOD                    => TO_PERIOD,
    WEEK_ARRAY                => FM_WEEK_ARRAY,
    STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                 STAFF_FRAME_BASE_RECORD)

   and then
```

```
STAFF_PACKAGE.
ALL_STAFF_FREE
(STAFF_LIST                => TO_ACTIVITY_PTR.
                              STAFF_LIST,
 MODULE                    => TO_ACTIVITY_PTR.
                              FACT,
 DAY                       => PERIOD_3_PACKAGE.
                              CONVERT(DAY => FM_DAY),
 PERIOD                    => FM_PERIOD,
 WEEK_ARRAY                => TO_WEEK_ARRAY,
 STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                              STAFF_FRAME_BASE_RECORD) then

TRY_SWAPPING_COMMON_MODULES
(COMMON_LIST       => FM_ACTIVITY_PTR.
                      LIST_OF_COMMON_ACTIVITIES,
 FM_ACTIVITY_PTR   => FM_ACTIVITY_PTR,
 FM_PERIOD_PTR     => FM_PERIOD_PTR,
 FM_DAY            => FM_DAY,
 FM_PERIOD         => FM_PERIOD,
 TO_ACTIVITY_PTR   => TO_ACTIVITY_PTR,
 TO_PERIOD_PTR     => TO_PERIOD_PTR,
 TO_DAY            => TO_DAY,
 TO_PERIOD         => TO_PERIOD,
 BLACKBOARD        => BLACKBOARD,
 SUCCESS           => LOCAL_SUCCESS);

if LOCAL_SUCCESS then

  SWAP_PERIODS
  (FM_PERIOD_PTR    => FM_PERIOD_PTR,
   FM_ACTIVITY_PTR => FM_ACTIVITY_PTR,
   FM_DAY           => FM_DAY,
   FM_PERIOD        => FM_PERIOD,
   TO_PERIOD_PTR    => TO_PERIOD_PTR,
   TO_ACTIVITY_PTR => TO_ACTIVITY_PTR,
   TO_DAY           => TO_DAY,
   TO_PERIOD        => TO_PERIOD,
   BLACKBOARD       => BLACKBOARD);

  ADJUST_KB
  (OLD_CLAUSE    => FM_TEMP_CLAUSE,
   ACTIVITY_PTR => FM_ACTIVITY_PTR,
   DEGREE        => DEGREE);

  ADJUST_KB
  (OLD_CLAUSE    => TO_TEMP_CLAUSE,
   ACTIVITY_PTR => FM_ACTIVITY_PTR,
   DEGREE        => DEGREE);

 STAFF_PACKAGE.
 MAKE_ALL_STAFF_FREE
 (STAFF_LIST               => FM_ACTIVITY_PTR.
                              STAFF_LIST,
  MODULE                   => FM_ACTIVITY_PTR.
                              FACT,
  DAY                      => PERIOD_3_PACKAGE.
                              CONVERT(DAY => FM_DAY),
  PERIOD                   => FM_PERIOD,
```

```
      WEEK_ARRAY                => FM_WEEK_ARRAY,
      STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                    STAFF_FRAME_BASE_RECORD);

    STAFF_PACKAGE.
    MAKE_ALL_STAFF_BUSY
    (STAFF_LIST               => FM_ACTIVITY_PTR.
                                    STAFF_LIST,
     MODULE                   => FM_ACTIVITY_PTR.
                                    FACT,
     DAY                      => PERIOD_3_PACKAGE.
                                    CONVERT(DAY => TO_DAY),
     PERIOD                   => TO_PERIOD,
     WEEK_ARRAY               => FM_WEEK_ARRAY,
     STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                    STAFF_FRAME_BASE_RECORD);

    STAFF_PACKAGE.
    MAKE_ALL_STAFF_FREE
    (STAFF_LIST               => TO_ACTIVITY_PTR.
                                    STAFF_LIST,
     MODULE                   => TO_ACTIVITY_PTR.
                                    FACT,
     DAY                      => PERIOD_3_PACKAGE.
                                    CONVERT(DAY => TO_DAY),
     PERIOD                   => TO_PERIOD,
     WEEK_ARRAY               => TO_WEEK_ARRAY,
     STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                    STAFF_FRAME_BASE_RECORD);

    STAFF_PACKAGE.
    MAKE_ALL_STAFF_BUSY
    (STAFF_LIST               => TO_ACTIVITY_PTR.
                                    STAFF_LIST,
     MODULE                   => TO_ACTIVITY_PTR.
                                    FACT,
     DAY                      => PERIOD_3_PACKAGE.
                                    CONVERT(DAY => FM_DAY),
     PERIOD                   => FM_PERIOD,
     WEEK_ARRAY               => TO_WEEK_ARRAY,
     STAFF_FRAME_BASE_RECORD => STAFF_PACKAGE.
                                    STAFF_FRAME_BASE_RECORD);

     SUCCESS := TRUE;

  else

     RESTORE_KB
     (CURRENT_CLAUSE => FM_TEMP_CLAUSE,
      OLD_CLAUSE     => FM_OLD_CLAUSE,
      DEGREE         => DEGREE);

     RESTORE_KB
     (CURRENT_CLAUSE => TO_TEMP_CLAUSE,
      OLD_CLAUSE     => TO_OLD_CLAUSE,
      DEGREE         => DEGREE);

     SUCCESS := FALSE;

  end if;
```

```
        else

          RESTORE_KB
          (CURRENT_CLAUSE => FM_TEMP_CLAUSE,
           OLD_CLAUSE     => FM_OLD_CLAUSE,
           DEGREE         => DEGREE);

          RESTORE_KB
          (CURRENT_CLAUSE => TO_TEMP_CLAUSE,
           OLD_CLAUSE     => TO_OLD_CLAUSE,
           DEGREE         => DEGREE);

          SUCCESS := FALSE;

        end if;
      end if;
    end if;
  ------------------------------------------------------------------
  exception
    when CONSTRAINT_ERROR =>
      TEXT_IO.PUT_LINE
      ("Exception CONSTRAINT_ERROR in SWAP_PACKAGE at " &
       "TRY_SWAPPING_PERIODS");
      SUCCESS := FALSE;
    when OTHERS =>
      TEXT_IO.PUT_LINE
      ("Exception OTHERS in SWAP_PACKAGE at " &
       "TRY_SWAPPING_PERIODS");
      SUCCESS := FALSE;
  ------------------------------------------------------------------
  end TRY_SWAPPING_PERIODS;
  ------------------------------------------------------------------

------------------------------------------------------------------
end SWAP_PACKAGE;
------------------------------------------------------------------
```

Window

```
-------------------------------------------------------------------
--
-- Unit      : WINDOW_PACKAGE specification
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 22 August 1993
-- Function  : This package provides the timetable window operations
--
-------------------------------------------------------------------
with TEXT_IO; use TEXT_IO;
with SYSTEM;
with OSUTILS;
with TOOLUTILS;
with UNCHECKED_CONVERSION;
with MAC_TYPES; use MAC_TYPES;
with QUICKDRAW; use QUICKDRAW;
with EVENTS;
with FONTS;
with MEMORY;
with MENUS;
with WINDOWS;
with CONTROLS; use CONTROLS;
with DESK;
with DIALOGS;
with TEXTEDIT;
with SYSTEM_TYPES_PACKAGE;
with TIMETABLE_TYPES_PACKAGE;
with TIMETABLE_KS_PACKAGE;
with TIMETABLE_BLACKBOARD_PACKAGE;
with MOVE_PACKAGE;
with SWAP_PACKAGE;
with TIMETABLE_SCHEDULER;
-------------------------------------------------------------------
package WINDOW_PACKAGE is
-------------------------------------------------------------------
  GBUFFER                       : TIMETABLE_KS_PACKAGE.
                                  CONSOLIDATED_BUFFER_TYPE;
  GDEGREE                       : TIMETABLE_KS_PACKAGE.DEGREE_TYPE :=
                                  TIMETABLE_KS_PACKAGE.IT_DEGREE;

  GWINDOW                       : QUICKDRAW.WINDOWPTR;
  GDONE                         : BOOLEAN := FALSE;
  GWNEINPLEMENTED               : BOOLEAN := FALSE;
  GEVENT                        : EVENTS.EVENTRECORD;

  GFONT_SIZE                    : constant := 9;
  GFONT                         : MAC_TYPES.INTEGER := FONTS.COURIER;
  GDROW                         : MAC_TYPES.INTEGER;
  GDCOL                         : MAC_TYPES.INTEGER;
  GROW                          : MAC_TYPES.INTEGER := 0;
  GCOL                          : MAC_TYPES.INTEGER := 0;

  GMENU_BAR                     : MAC_TYPES.HANDLE;

  GFILE_MENU_ID                 : constant := 400;
  GPRINT                        : constant := 1;
  GQUIT                         : constant := 3;
```

```
  GCHANGE_MENU_ID                : constant := 401;
  GMOVE                          : constant := 1;
  GSWAP                          : constant := 2;

  GDEGREE_MENU_ID                : constant := 402;
  GIT                            : constant := 1;
  GISE                           : constant := 2;
  GESE                           : constant := 3;
  GCIS                           : constant := 4;
  GISM                           : constant := 5;

  GDIALOG                        : DIALOGS.DIALOGPTR;
  GFROM_DIALOG_ID                : constant := 400;
  GTO_DIALOG_ID                  : constant := 401;
  GNAME_DIALOG_ID                : constant := 402;
  GMOVE_NOT_POSSIBLE_DIALOG_ID : constant := 403;
  GSWAP_NOT_POSSIBLE_DIALOG_ID : constant := 404;

  FUNCTION ADDRESS_TO_PTR IS new UNCHECKED_CONVERSION
  (SOURCE => SYSTEM.ADDRESS,
   TARGET => MAC_TYPES.PTR);

  FUNCTION LONGINT_TO_WINDOWPTR IS new UNCHECKED_CONVERSION
  (SOURCE => MAC_TYPES.LONGINT,
   TARGET => QUICKDRAW.WINDOWPTR);

  FUNCTION ADDRESS_TO_VARWINDOWPTR IS new UNCHECKED_CONVERSION
  (SOURCE => SYSTEM.ADDRESS,
   TARGET => QUICKDRAW.VARWINDOWPTR);

  FUNCTION ADDRESS_TO_PROCPTR IS new UNCHECKED_CONVERSION
  (SOURCE => SYSTEM.ADDRESS,
   TARGET => MAC_TYPES.PROCPTR);

  FUNCTION HANDLE_TO_CONTROL_HANDLE is new UNCHECKED_CONVERSION
  (SOURCE => MAC_TYPES.HANDLE,
   TARGET => CONTROLS.CONTROLHANDLE);

  FUNCTION DIALOGPTR_TO_PTR is new UNCHECKED_CONVERSION
  (SOURCE => DIALOGS.DIALOGPEEK,
   TARGET => MAC_TYPES.PTR);

  procedure INITIALISE_WINDOW;
  procedure HANDLE_EVENT;
  procedure DRIVER;

------------------------------------------------------------------------
end WINDOW_PACKAGE;
------------------------------------------------------------------------
```

```
------------------------------------------------------------------------
--
-- Unit      : WINDOW_PACKAGE body
-- Author    : A Harrison, Software Engineering Group,
--             Cranfield University, RMCS, Shrivenham
-- Date      : 22 August 1993
-- Function : This package provides the timetable window operations
--
------------------------------------------------------------------------
package body WINDOW_PACKAGE is
------------------------------------------------------------------------
  ----------------------------------------------------------------------
  procedure INITIALISE_ERROR_DIALOG
  ----------------------------------------------------------------------
  (DIALOG_ID  : in     MAC_TYPES.INTEGER;
   THE_DIALOG : in out DIALOGS.DIALOGPTR) is
  ----------------------------------------------------------------------
  begin

    THE_DIALOG := DIALOGS.GETNEWDIALOG
                  (DIALOGID => DIALOG_ID,
                   DSTORAGE => null,
                   BEHIND   => LONGINT_TO_WINDOWPTR(-1));
  ----------------------------------------------------------------------
  end INITIALISE_ERROR_DIALOG;
  ----------------------------------------------------------------------


  ----------------------------------------------------------------------
  procedure ERROR_DIALOG
  ----------------------------------------------------------------------
  (THE_DIALOG    : in out DIALOGS.DIALOGPTR) is
  ----------------------------------------------------------------------
    DIALOG_DONE     : BOOLEAN := FALSE;
    ITEM_HIT        : MAC_TYPES.INTEGER := 0;
    ITEM_TYPE       : MAC_TYPES.INTEGER;
    ITEM_RECT       : MAC_TYPES.RECT;
    ITEM_HANDLE     : MAC_TYPES.HANDLE;

    OK   : constant := 1;
  ----------------------------------------------------------------------
  begin

    WINDOWS.SHOWWINDOW(THEWINDOW => THE_DIALOG);

    while not DIALOG_DONE
    loop

      DIALOGS.MODALDIALOG
              (FILTERPROC => null,
               ITEMHIT    => ITEM_HIT);

      case ITEM_HIT is

        when OK       => WINDOWS.
                         SHOWWINDOW(THEWINDOW => THE_DIALOG);
                         MENUS.HILITEMENU(0);
                         DIALOG_DONE := TRUE;
```

```
      when others   => null;
    end case;
  end loop;
-------------------------------------------------------------------
end ERROR_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure INITIALISE_NAME_SELECT_DIALOG
(DIALOG_ID  : in     MAC_TYPES.INTEGER;
 THE_DIALOG : in out DIALOGS.DIALOGPTR) is
-------------------------------------------------------------------
begin

  THE_DIALOG := DIALOGS.GETNEWDIALOG
                (DIALOGID => DIALOG_ID,
                 DSTORAGE => null,
                 BEHIND   => LONGINT_TO_WINDOWPTR(-1));
-------------------------------------------------------------------
end INITIALISE_NAME_SELECT_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure NAME_SELECT_DIALOG
-------------------------------------------------------------------
(NAME         : in out MAC_TYPES.STR255;
 THE_DIALOG   : in out DIALOGS.DIALOGPTR) is
-------------------------------------------------------------------
  DIALOG_DONE     : BOOLEAN := FALSE;
  ITEM_HIT        : MAC_TYPES.INTEGER := 0;
  ITEM_TYPE       : MAC_TYPES.INTEGER;
  ITEM_RECT       : MAC_TYPES.RECT;
  ITEM_HANDLE     : MAC_TYPES.HANDLE;

  TEXT : constant := 1;
  OK   : constant := 3;
-------------------------------------------------------------------
begin
  WINDOWS.SHOWWINDOW(THEWINDOW => THE_DIALOG);

  while not DIALOG_DONE
  loop

    DIALOGS.MODALDIALOG
            (FILTERPROC => null,
             ITEMHIT    => ITEM_HIT);

    case ITEM_HIT is

      when TEXT  => -- Get text

                    DIALOGS.
                    GETDITEM
                    (THEDIALOG => THE_DIALOG,
                     ITEMNO    => TEXT,
                     ITEMTYPE  => ITEM_TYPE,
                     ITEM      => ITEM_HANDLE,
                     BOX       => ITEM_RECT);
```

```
                    DIALOGS.
                    GETITEXT
                    (ITEM => ITEM_HANDLE,
                     TEXT => NAME);

      when OK     => WINDOWS.SHOWWINDOW(THEWINDOW => THE_DIALOG);
                     MENUS.HILITEMENU(0);
                     DIALOG_DONE := TRUE;

      when others => null;
    end case;
  end loop;
-------------------------------------------------------------------
end NAME_SELECT_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_NAME_SELECT_DIALOG
(NAME: in out MAC_TYPES.STR255) is
-------------------------------------------------------------------
begin

  INITIALISE_NAME_SELECT_DIALOG
  (DIALOG_ID  => GNAME_DIALOG_ID,
   THE_DIALOG => GDIALOG);

  NAME_SELECT_DIALOG
  (NAME       => NAME,
   THE_DIALOG => GDIALOG);

  DIALOGS.DISPOSEDIALOG
  (THEDIALOG => GDIALOG);
-------------------------------------------------------------------
end HANDLE_NAME_SELECT_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure INITIALISE_PERIOD_SELECT_DIALOG
-------------------------------------------------------------------
(DIALOG_ID  : in     MAC_TYPES.INTEGER;
 THE_DIALOG : in out DIALOGS.DIALOGPTR) is
-------------------------------------------------------------------
  ITEM_TYPE   : MAC_TYPES.INTEGER;
  ITEM_RECT   : MAC_TYPES.RECT;
  ITEM_HANDLE : MAC_TYPES.HANDLE;

  subtype DIALOG_RANGE is MAC_TYPES.INTEGER range 6 .. 17;
  ON  : constant MAC_TYPES.INTEGER := 1;
  OFF : constant MAC_TYPES.INTEGER := 0;
  CONTROL_HANDLE : CONTROLS.CONTROLHANDLE;
-------------------------------------------------------------------
begin
  THE_DIALOG := DIALOGS.GETNEWDIALOG
                (DIALOGID => DIALOG_ID,
                 DSTORAGE => null,
                 BEHIND   => LONGINT_TO_WINDOWPTR(-1));
```

```
  for ITEM in DIALOG_RANGE
  loop
    DIALOGS.GETDITEM
            (THEDIALOG => THE_DIALOG,
             ITEMNO    => ITEM,
             ITEMTYPE  => ITEM_TYPE,
             ITEM      => ITEM_HANDLE,
             BOX       => ITEM_RECT);

    CONTROL_HANDLE := HANDLE_TO_CONTROL_HANDLE(ITEM_HANDLE);

    CONTROLS.SETCTLVALUE
             (THECONTROL => CONTROL_HANDLE,
              THEVALUE   => OFF);
  end loop;
-------------------------------------------------------------------
end INITIALISE_PERIOD_SELECT_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure PERIOD_SELECT_DIALOG
-------------------------------------------------------------------
  (DAY           :    out TIMETABLE_TYPES_PACKAGE.
                          TIMETABLE_ITEM_TYPE;
   PERIOD        :    out TIMETABLE_TYPES_PACKAGE.
                         PERIOD_NUMBER_TYPE;
   THE_DIALOG    : in out DIALOGS.DIALOGPTR) is
-------------------------------------------------------------------
  DIALOG_DONE     : BOOLEAN := FALSE;
  ITEM_HIT        : MAC_TYPES.INTEGER := 0;
  ITEM_TYPE       : MAC_TYPES.INTEGER;
  ITEM_RECT       : MAC_TYPES.RECT;
  ITEM_HANDLE     : MAC_TYPES.HANDLE;
  SELECTED_DAY    : MAC_TYPES.INTEGER := 0;
  SELECTED_PERIOD : MAC_TYPES.INTEGER := 0;

  OK  : constant := 2;
  MON : constant := 6;
  TUE : constant := 7;
  WED : constant := 8;
  THU : constant := 9;
  FRI : constant := 10;
  P1  : constant := 11;
  P2  : constant := 12;
  P3  : constant := 13;
  P4  : constant := 14;
  P5  : constant := 15;
  P6  : constant := 16;
  P7  : constant := 17;
  ON  : constant := 1;
  OFF : constant := 0;
  -----------------------------------------------------------------
  procedure SET_ITEM
  -----------------------------------------------------------------
  (SELECTION : in out MAC_TYPES.INTEGER;
   ITEM      : in     MAC_TYPES.INTEGER;
   DIALOG    : in out DIALOGS.DIALOGPTR) is
  -----------------------------------------------------------------
  begin
```

```
      if SELECTION /= 0 -- delete it
      then
        -- Get old handle and turn it off
        DIALOGS.GETDITEM
                (THEDIALOG => DIALOG,
                 ITEMNO    => SELECTION,
                 ITEMTYPE  => ITEM_TYPE,
                 ITEM      => ITEM_HANDLE,
                 BOX       => ITEM_RECT);
         CONTROLS.
         SETCTLVALUE
         (THECONTROL => HANDLE_TO_CONTROL_HANDLE(ITEM_HANDLE),
          THEVALUE   => OFF);
      end if;

      -- Get new handle and turn it on

      DIALOGS.GETDITEM
              (THEDIALOG => DIALOG,
               ITEMNO    => ITEM,
               ITEMTYPE  => ITEM_TYPE,
               ITEM      => ITEM_HANDLE,
               BOX       => ITEM_RECT);
       CONTROLS.SETCTLVALUE
               (THECONTROL => HANDLE_TO_CONTROL_HANDLE
                              (ITEM_HANDLE),
                THEVALUE   => ON);
      -- Record new selection
      SELECTION := ITEM;
  ------------------------------------------------------------------
    end SET_ITEM;
------------------------------------------------------------------
begin
  WINDOWS.SHOWWINDOW(THEWINDOW => THE_DIALOG);
  while not DIALOG_DONE
  loop
    DIALOGS.MODALDIALOG
            (FILTERPROC => null,
             ITEMHIT    => ITEM_HIT);
    case ITEM_HIT is
      when MON|TUE|WED|THU|FRI    => SET_ITEM
                                     (SELECTION => SELECTED_DAY,
                                      ITEM      => ITEM_HIT,
                                      DIALOG    => THE_DIALOG);
       when P1|P2|P3|P4|P5|P6|P7  => SET_ITEM
                                     (SELECTION => SELECTED_PERIOD,
                                      ITEM      => ITEM_HIT,
                                      DIALOG    => THE_DIALOG);
       when OK                    => if SELECTED_DAY /= 0 and
                                        SELECTED_PERIOD /= 0
                                     then
                                       WINDOWS.
                                       SHOWWINDOW
                                       (THEWINDOW => THE_DIALOG);
                                       MENUS.HILITEMENU(0);
                                       DIALOGS.
                                       GETDITEM
                                       (THEDIALOG => THE_DIALOG,
                                        ITEMNO    => SELECTED_DAY,
                                        ITEMTYPE  => ITEM_TYPE,
```

```
                                    ITEM     => ITEM_HANDLE,
                                    BOX      => ITEM_RECT);
                                  CONTROLS.SETCTLVALUE
                                  (THECONTROL =>
                                   HANDLE_TO_CONTROL_HANDLE
                                   (ITEM_HANDLE),
                                   THEVALUE   => OFF);
                                  DIALOGS.GETDITEM
                                  (THEDIALOG => THE_DIALOG,
                                   ITEMNO    =>
                                   SELECTED_PERIOD,
                                   ITEMTYPE  => ITEM_TYPE,
                                   ITEM      => ITEM_HANDLE,
                                   BOX       => ITEM_RECT);
                                  CONTROLS.SETCTLVALUE
                                  (THECONTROL =>
                                   HANDLE_TO_CONTROL_HANDLE
                                   (ITEM_HANDLE),
                                   THEVALUE   => OFF);

                                  case SELECTED_DAY is
                                    when MON => DAY :=
                                      TIMETABLE_TYPES_PACKAGE.
                                      MONDAY;
                                    when TUE => DAY :=
                                      TIMETABLE_TYPES_PACKAGE.
                                      TUESDAY;
                                    when WED => DAY :=
                                      TIMETABLE_TYPES_PACKAGE.
                                      WEDNESDAY;
                                    when THU => DAY :=
                                      TIMETABLE_TYPES_PACKAGE.
                                      THURSDAY;
                                    when FRI => DAY :=
                                      TIMETABLE_TYPES_PACKAGE.
                                      FRIDAY;
                                    when others => null;
                                  end case;
                                  case SELECTED_PERIOD is
                                    when P1 => PERIOD := 1;
                                    when P2 => PERIOD := 2;
                                    when P3 => PERIOD := 3;
                                    when P4 => PERIOD := 4;
                                    when P5 => PERIOD := 5;
                                    when P6 => PERIOD := 6;
                                    when P7 => PERIOD := 7;
                                    when others => null;
                                  end case;
                                  DIALOG_DONE := TRUE;
                                 else
                                   null;
                                 end if;

      when others                => null;
    end case;
  end loop;
-------------------------------------------------------------------
end PERIOD_SELECT_DIALOG;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure INITIALISE_WINDOW is
-------------------------------------------------------------------
  WINDOW_ID     : constant := 128;
-------------------------------------------------------------------
begin

  QUICKDRAW.
  INITGRAF
  (GLOBALPTR => ADDRESS_TO_PTR(QUICKDRAW.QD.THEPORT'ADDRESS));
  FONTS.INITFONTS;
  WINDOWS.INITWINDOWS;
  MENUS.INITMENUS;
  TEXTEDIT.TEINIT;
  DIALOGS.INITDIALOGS(null);
  QUICKDRAW.INITCURSOR;

  GWINDOW := WINDOWS.
             GETNEWCWINDOW
             (WINDOWID => WINDOW_ID,
              WSTORAGE => NULL,
              BEHIND   => LONGINT_TO_WINDOWPTR(-1));

  WINDOWS.SELECTWINDOW(GWINDOW);
  WINDOWS.SHOWWINDOW(GWINDOW);
  QUICKDRAW.SETPORT(GWINDOW);
-------------------------------------------------------------------
end INITIALISE_WINDOW;
-------------------------------------------------------------------

-------------------------------------------------------------------
procedure INITIALISE_MENU_BAR is
-------------------------------------------------------------------
  MY_MENU_BAR : MAC_TYPES.HANDLE;
  A_MENU      : MENUS.MENUHANDLE;
  MENU_BAR_ID : constant MAC_TYPES.INTEGER := 400;
-------------------------------------------------------------------
begin

  -- Clear existing menubar

  GMENU_BAR := MENUS.GETMENUBAR;
  MENUS.CLEARMENUBAR;
  MENUS.DRAWMENUBAR;

  -- Set new menubar

  MY_MENU_BAR := MENUS.GETNEWMBAR(MENUBARID => MENU_BAR_ID);

  if MY_MENU_BAR /= null then

    MENUS.SETMENUBAR(MENULIST => MY_MENU_BAR);
    MEMORY.DISPOSEHANDLE(H => MY_MENU_BAR);

    MENUS.DRAWMENUBAR;

  end if;
-------------------------------------------------------------------
end INITIALISE_MENU_BAR;
```

```
-------------------------------------------------------------------


-------------------------------------------------------------------
function STRING_TO_STR255
-------------------------------------------------------------------
(ADA : in STANDARD.STRING) return MAC_TYPES.STR255 is
-------------------------------------------------------------------
  STR : MAC_TYPES.STR255;
-------------------------------------------------------------------
begin
  for I in 1 .. ADA'LENGTH
  loop
    STR(I) := MAC_TYPES.CHAR'VAL(CHARACTER'POS(ADA(I)));
  end loop;
  STR(0) := MAC_TYPES.CHAR'VAL(ADA'LENGTH);
  return STR;
-------------------------------------------------------------------
end STRING_TO_STR255;
-------------------------------------------------------------------


-------------------------------------------------------------------
function STR255_TO_STRING
-------------------------------------------------------------------
(STR : in MAC_TYPES.STR255) return STANDARD.STRING is
-------------------------------------------------------------------
  THE_STRING : STANDARD.STRING(1 .. 20);
  LENGTH     : NATURAL;
-------------------------------------------------------------------
begin
  LENGTH := MAC_TYPES.CHAR'POS(STR(0));
  if LENGTH = 0 then
    return "";
  else
    for I in 1 .. LENGTH
    loop
      THE_STRING(I) := CHARACTER'VAL(CHAR'POS(STR(I)));
    end loop;
    return THE_STRING(1 .. LENGTH);
  end if;
-------------------------------------------------------------------
end STR255_TO_STRING;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SET_DRAWING_PARAMETERS is
-------------------------------------------------------------------
  FONT_INFO : QUICKDRAW.FONTINFO;
-------------------------------------------------------------------
begin
  QUICKDRAW.GETFONTINFO(FONT_INFO);
  GDROW := FONT_INFO.ASCENT +
           FONT_INFO.DESCENT +
           FONT_INFO.LEADING;
  GDCOL := QUICKDRAW.CHARWIDTH(MAC_TYPES.'X');
-------------------------------------------------------------------
end SET_DRAWING_PARAMETERS;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure MOVE_TO
-------------------------------------------------------------------
( ROW : in MAC_TYPES.INTEGER;
  COL : in MAC_TYPES.INTEGER) is
-------------------------------------------------------------------
begin
  GROW := GDROW * ROW;
  GCOL := GDCOL * COL;
  QUICKDRAW.MOVETO(GCOL, GROW);
-------------------------------------------------------------------
end MOVE_TO;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure LINE_TO
-------------------------------------------------------------------
( ROW : in MAC_TYPES.INTEGER;
  COL : in MAC_TYPES.INTEGER) is
-------------------------------------------------------------------
begin
  GROW := GDROW * ROW;
  GCOL := GDCOL * COL;
  QUICKDRAW.LINETO(GCOL, GROW);
-------------------------------------------------------------------
end LINE_TO;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure DRAW_GRID is
-------------------------------------------------------------------
  RIGHT      : MAC_TYPES.INTEGER := 126;
  LEFT       : MAC_TYPES.INTEGER := 1;
  TOP        : MAC_TYPES.INTEGER := 1;
  BOTTOM     : MAC_TYPES.INTEGER := 64;
-------------------------------------------------------------------
begin
  -- Out side box
  MOVE_TO(TOP,LEFT);
  LINE_TO(TOP,RIGHT);
  LINE_TO(BOTTOM,RIGHT);
  LINE_TO(BOTTOM,LEFT);
  LINE_TO(TOP,LEFT);

  -- Horizontal lines

  MOVE_TO(4,LEFT);
  LINE_TO(4,RIGHT);
  MOVE_TO(16,LEFT);
  LINE_TO(16,RIGHT);
  MOVE_TO(28,LEFT);
  LINE_TO(28,RIGHT);
  MOVE_TO(40,LEFT);
  LINE_TO(40,RIGHT);
  MOVE_TO(52,LEFT);
  LINE_TO(52,RIGHT);
```

```
-- Vertical lines

MOVE_TO(TOP,7);
LINE_TO(BOTTOM,7);
MOVE_TO(TOP,24);
LINE_TO(BOTTOM,24);
MOVE_TO(TOP,41);
LINE_TO(BOTTOM,41);
MOVE_TO(TOP,58);
LINE_TO(BOTTOM,58);
MOVE_TO(TOP,75);
LINE_TO(BOTTOM,75);
MOVE_TO(TOP,92);
LINE_TO(BOTTOM,92);
MOVE_TO(TOP,109);
LINE_TO(BOTTOM,109);

-- Time/day

MOVE_TO(2,2);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("TIME/"));
MOVE_TO(3,2);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("DAY"));

--  Period times

MOVE_TO(2,14);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("0850"));
MOVE_TO(2,31);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("0950"));
MOVE_TO(2,48);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1110"));
MOVE_TO(2,65);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1210"));
MOVE_TO(2,82);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1410"));
MOVE_TO(2,99);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1510"));
MOVE_TO(2,116);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1610"));

MOVE_TO(3,14);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("0940"));
MOVE_TO(3,31);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1040"));
MOVE_TO(3,48);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1200"));
MOVE_TO(3,65);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1300"));
MOVE_TO(3,82);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1500"));
MOVE_TO(3,99);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1600"));
MOVE_TO(3,116);
QUICKDRAW.DRAWSTRING(STRING_TO_STR255("1700"));

-- Days

MOVE_TO(10,3);
```

```
  QUICKDRAW.DRAWSTRING(STRING_TO_STR255("MON"));
  MOVE_TO(22,3);
  QUICKDRAW.DRAWSTRING(STRING_TO_STR255("TUE"));
  MOVE_TO(34,3);
  QUICKDRAW.DRAWSTRING(STRING_TO_STR255("WED"));
  MOVE_TO(46,3);
  QUICKDRAW.DRAWSTRING(STRING_TO_STR255("THU"));
  MOVE_TO(58,3);
  QUICKDRAW.DRAWSTRING(STRING_TO_STR255("FRI"));
-------------------------------------------------------------------
end DRAW_GRID;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure DRAW_TEXT(DEGREE : in TIMETABLE_KS_PACKAGE.
                                DEGREE_TYPE) is
-------------------------------------------------------------------
  ROW_DATUM : MAC_TYPES.INTEGER := 5;
  ROW       : MAC_TYPES.INTEGER := 5;
  COL       : MAC_TYPES.INTEGER := 8;
-------------------------------------------------------------------
begin
  TIMETABLE_KS_PACKAGE.GET_BUFFER(BUFFER => GBUFFER);
  for PERIOD in TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE
  loop
    declare
      THIS_PERIOD : TIMETABLE_KS_PACKAGE. -
                    CONSOLIDATED_PERIOD_TYPE :=
                    GBUFFER(PERIOD);
    begin
    for DAY in TIMETABLE_TYPES_PACKAGE.MONDAY ..
               TIMETABLE_TYPES_PACKAGE.FRIDAY
    loop
      ROW := ROW_DATUM;
      for GROUP in 1..10
      loop
        for ITEM in 1..20
        loop
          if not SYSTEM_TYPES_PACKAGE.
                 DYNAMIC_STRING.
                 IS_NULL
                 (THE_STRING => THIS_PERIOD(GROUP, DAY, DEGREE)
                                              (ITEM)) then
            MOVE_TO(ROW, COL);
            QUICKDRAW.DRAWSTRING
            (STRING_TO_STR255(SYSTEM_TYPES_PACKAGE.
                              DYNAMIC_STRING.
                              SUBSTRING_OF
                              (THIS_PERIOD(GROUP, DAY, DEGREE)
                                          (ITEM))));
            ROW := ROW + 1;
          end if;
        end loop;                        -- Item
      end loop;                          -- Group
      ROW_DATUM := ROW_DATUM + 12;
    end loop;                            -- Day
    ROW_DATUM := 5;
    COL := COL + 17;
    end;
```

```
  end loop;                                -- Period
-------------------------------------------------------------------
end DRAW_TEXT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_ERROR_DIALOG
-------------------------------------------------------------------
(DIALOG_ID : in MAC_TYPES.INTEGER) is
-------------------------------------------------------------------
begin

  INITIALISE_ERROR_DIALOG
  (DIALOG_ID  => DIALOG_ID,
   THE_DIALOG => GDIALOG);

  ERROR_DIALOG
  (THE_DIALOG => GDIALOG);

  DIALOGS.DISPOSEDIALOG
  (THEDIALOG => GDIALOG);

  DRAW_GRID;
  DRAW_TEXT(DEGREE => GDEGREE);
-------------------------------------------------------------------
end HANDLE_ERROR_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_PERIOD_SWAP_DIALOG
-------------------------------------------------------------------
(FM_DAY    :  in out TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
 FM_PERIOD :  in out TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
 FM_NAME   :  in out MAC_TYPES.STR255;
 TO_DAY    :  in out TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
 TO_PERIOD :  in out TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
 TO_NAME   :  in out MAC_TYPES.STR255 ) is
-------------------------------------------------------------------
begin

  INITIALISE_PERIOD_SELECT_DIALOG
  (DIALOG_ID  => GFROM_DIALOG_ID,
   THE_DIALOG => GDIALOG);

  PERIOD_SELECT_DIALOG
  (DAY        => FM_DAY,
   PERIOD     => FM_PERIOD,
   THE_DIALOG => GDIALOG);

  DIALOGS.DISPOSEDIALOG
  (THEDIALOG => GDIALOG);

  DRAW_GRID;
  DRAW_TEXT(DEGREE => GDEGREE);

  INITIALISE_NAME_SELECT_DIALOG
  (DIALOG_ID  => GNAME_DIALOG_ID,
   THE_DIALOG => GDIALOG);
```

```
   NAME_SELECT_DIALOG
   (NAME        => FM_NAME,
    THE_DIALOG => GDIALOG);

   DIALOGS.DISPOSEDIALOG
   (THEDIALOG => GDIALOG);

   DRAW_GRID;
   DRAW_TEXT(DEGREE => GDEGREE);

   INITIALISE_PERIOD_SELECT_DIALOG
   (DIALOG_ID  => GTO_DIALOG_ID,
    THE_DIALOG => GDIALOG);

   PERIOD_SELECT_DIALOG
   (DAY         => TO_DAY,
    PERIOD      => TO_PERIOD,
    THE_DIALOG => GDIALOG);

   DIALOGS.DISPOSEDIALOG
   (THEDIALOG => GDIALOG);

   DRAW_GRID;
   DRAW_TEXT(DEGREE => GDEGREE);

   INITIALISE_NAME_SELECT_DIALOG
   (DIALOG_ID  => GNAME_DIALOG_ID,
    THE_DIALOG => GDIALOG);

   NAME_SELECT_DIALOG
   (NAME        => TO_NAME,
    THE_DIALOG => GDIALOG);

   DIALOGS.DISPOSEDIALOG
   (THEDIALOG => GDIALOG);

   DRAW_GRID;
   DRAW_TEXT(DEGREE => GDEGREE);
-------------------------------------------------------------------
end HANDLE_PERIOD_SWAP_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_PERIOD_MOVE_DIALOG
-------------------------------------------------------------------
(FM_DAY    :  in out TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
 FM_PERIOD :  in out TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
 FM_NAME   :  in out MAC_TYPES.STR255;
 TO_DAY    :  in out TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
 TO_PERIOD :  in out TIMETABLE_TYPES_PACKAGE.
                     PERIOD_NUMBER_TYPE ) is
-------------------------------------------------------------------
begin

   INITIALISE_PERIOD_SELECT_DIALOG
   (DIALOG_ID  => GFROM_DIALOG_ID,
    THE_DIALOG => GDIALOG);
```

```
   PERIOD_SELECT_DIALOG
   (DAY         => FM_DAY,
    PERIOD      => FM_PERIOD,
    THE_DIALOG => GDIALOG);

   DIALOGS.DISPOSEDIALOG
   (THEDIALOG => GDIALOG);

   DRAW_GRID;
   DRAW_TEXT(DEGREE => GDEGREE);

   INITIALISE_NAME_SELECT_DIALOG
   (DIALOG_ID  => GNAME_DIALOG_ID,
    THE_DIALOG => GDIALOG);

   NAME_SELECT_DIALOG
   (NAME        => FM_NAME,
    THE_DIALOG => GDIALOG);

   DIALOGS.DISPOSEDIALOG
   (THEDIALOG => GDIALOG);

   DRAW_GRID;
   DRAW_TEXT(DEGREE => GDEGREE);

   INITIALISE_PERIOD_SELECT_DIALOG
   (DIALOG_ID  => GTO_DIALOG_ID,
    THE_DIALOG => GDIALOG);

   PERIOD_SELECT_DIALOG
   (DAY         => TO_DAY,
    PERIOD      => TO_PERIOD,
    THE_DIALOG => GDIALOG);

   DIALOGS.DISPOSEDIALOG
   (THEDIALOG => GDIALOG);

   DRAW_GRID;
   DRAW_TEXT(DEGREE => GDEGREE);

end HANDLE_PERIOD_MOVE_DIALOG;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure SHOW
-------------------------------------------------------------------
(DEGREE : in TIMETABLE_KS_PACKAGE.DEGREE_TYPE) is
-------------------------------------------------------------------
begin

   -- Clear current display

   QUICKDRAW.FORECOLOR(COLOR => QUICKDRAW.WHITECOLOR);
   DRAW_TEXT(GDEGREE);

   -- Change degree

   GDEGREE := DEGREE;
```

```
   -- Show new degree

   QUICKDRAW.FORECOLOR(COLOR => QUICKDRAW.BLACKCOLOR);
   DRAW_TEXT(GDEGREE);

-------------------------------------------------------------------
end SHOW;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_DEGREE_CHOICE
(THE_ITEM : in MAC_TYPES.INTEGER) is
begin
  case THE_ITEM is
    when GIT    => SHOW(DEGREE => TIMETABLE_KS_PACKAGE.
                                  IT_DEGREE);
    when GISE   => null;
    when GESE   => SHOW(DEGREE => TIMETABLE_KS_PACKAGE.
                                  ESE_DEGREE);
    when GCIS   => null;
    when GISM   => null;
    when others => null;
  end case;
end HANDLE_DEGREE_CHOICE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_FILE_CHOICE
-------------------------------------------------------------------

(THE_ITEM : in MAC_TYPES.INTEGER) is
-------------------------------------------------------------------
begin
  case THE_ITEM is
    when GPRINT  => null;
    when GQUIT   => WINDOWS.DISPOSEWINDOW(GWINDOW);
                    MENUS.CLEARMENUBAR;
                    MENUS.DRAWMENUBAR;
                    MENUS.SETMENUBAR(MENULIST => GMENU_BAR);
                    MENUS.DRAWMENUBAR;
                    GDONE := TRUE;
    when others => null;
  end case;
-------------------------------------------------------------------
end HANDLE_FILE_CHOICE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_CHANGE_CHOICE
-------------------------------------------------------------------
(THE_ITEM : in MAC_TYPES.INTEGER) is
-------------------------------------------------------------------
  FM_DAY         : TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
  FM_PERIOD      : TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
  FM_MODULE_NAME : MAC_TYPES.STR255;
  TO_DAY         : TIMETABLE_TYPES_PACKAGE.TIMETABLE_ITEM_TYPE;
  TO_PERIOD      : TIMETABLE_TYPES_PACKAGE.PERIOD_NUMBER_TYPE;
```

```
  TO_MODULE_NAME : MAC_TYPES.STR255;

  SUCCESS : BOOLEAN := FALSE;
-------------------------------------------------------------------
begin

  case THE_ITEM is

    when GMOVE  => HANDLE_PERIOD_MOVE_DIALOG
                   (FM_DAY    => FM_DAY,
                    FM_PERIOD => FM_PERIOD,
                    FM_NAME   => FM_MODULE_NAME,
                    TO_DAY    => TO_DAY,
                    TO_PERIOD => TO_PERIOD);

                    -- Change the blackboard

                    MOVE_PACKAGE.
                    MOVE_TO
                    (FM_ACTIVITY_NAME    =>
                     STR255_TO_STRING(STR => FM_MODULE_NAME),
                     FM_DAY              => FM_DAY,
                     FM_PERIOD           => FM_PERIOD,
                     TO_DAY              => TO_DAY,
                     TO_PERIOD           => TO_PERIOD,
                     BLACKBOARD          =>
                     TIMETABLE_BLACKBOARD_PACKAGE.
                     BLACKBOARD,
                     SUCCESS             => SUCCESS);

                   if SUCCESS then

                    -- Clear the text

                    QUICKDRAW.FORECOLOR
                    (COLOR => QUICKDRAW.WHITECOLOR);
                    DRAW_TEXT(GDEGREE);
                    QUICKDRAW.FORECOLOR
                    (COLOR => QUICKDRAW.BLACKCOLOR);

                     -- Produce new timetable

                     TIMETABLE_KS_PACKAGE.
                     PROCESS_EVENT
                     (BLACKBOARD => TIMETABLE_BLACKBOARD_PACKAGE.
                                    BLACKBOARD);

                     -- Redraw the text

                     DRAW_TEXT(GDEGREE);

                   else

                     -- Handle error dialog

                     HANDLE_ERROR_DIALOG
                     (DIALOG_ID => GMOVE_NOT_POSSIBLE_DIALOG_ID);

                   end if;
```

```
    when GSWAP  => HANDLE_PERIOD_SWAP_DIALOG
                   (FM_DAY    => FM_DAY,
                    FM_PERIOD => FM_PERIOD,
                    FM_NAME   => FM_MODULE_NAME,
                    TO_DAY    => TO_DAY,
                    TO_PERIOD => TO_PERIOD,
                    TO_NAME   => TO_MODULE_NAME);

                   -- Change the blackboard

                   SWAP_PACKAGE.
                   TRY_SWAPPING_PERIODS
                   (FM_ACTIVITY_NAME    =>
                    STR255_TO_STRING(STR => FM_MODULE_NAME),
                    FM_DAY              => FM_DAY,
                    FM_PERIOD           => FM_PERIOD,
                    TO_ACTIVITY_NAME    =>
                    STR255_TO_STRING(STR => TO_MODULE_NAME),
                    TO_DAY              => TO_DAY,
                    TO_PERIOD           => TO_PERIOD,
                    BLACKBOARD          =>
                    TIMETABLE_BLACKBOARD_PACKAGE.
                    BLACKBOARD,
                    SUCCESS             => SUCCESS);

                   if SUCCESS then

                     -- Clear the text

                     QUICKDRAW.FORECOLOR
                     (COLOR => QUICKDRAW.WHITECOLOR);
                     DRAW_TEXT(GDEGREE);
                     QUICKDRAW.FORECOLOR
                     (COLOR => QUICKDRAW.BLACKCOLOR);

                     -- Produce new timetable

                     TIMETABLE_KS_PACKAGE.
                     PROCESS_EVENT
                     (BLACKBOARD => TIMETABLE_BLACKBOARD_PACKAGE.
                                    BLACKBOARD);

                     -- Redraw the text

                     DRAW_TEXT(GDEGREE);

                   else

                     -- Handle error dialog

                     HANDLE_ERROR_DIALOG
                     (DIALOG_ID => GSWAP_NOT_POSSIBLE_DIALOG_ID);

                   end if;

    when others   => null;
  end case;
-------------------------------------------------------------------
end HANDLE_CHANGE_CHOICE;
-------------------------------------------------------------------
```

```
-------------------------------------------------------------------
procedure HANDLE_MENU_CHOICE
-------------------------------------------------------------------
(SELECTED : in MAC_TYPES.LONGINT) is
-------------------------------------------------------------------
  THE_MENU : MAC_TYPES.INTEGER;
  THE_ITEM : MAC_TYPES.INTEGER;
-------------------------------------------------------------------
begin
  if  SELECTED /= 0 then
    THE_MENU := TOOLUTILS.HIWORD(SELECTED);
    THE_ITEM := TOOLUTILS.LOWORD(SELECTED);
    case THE_MENU is
      when  GFILE_MENU_ID   => HANDLE_FILE_CHOICE
                               (THE_ITEM   => THE_ITEM);
      when  GCHANGE_MENU_ID => HANDLE_CHANGE_CHOICE
                               (THE_ITEM => THE_ITEM);
      when  GDEGREE_MENU_ID => HANDLE_DEGREE_CHOICE
                               (THE_ITEM => THE_ITEM);
      when others           => null;
    end case;
  end if;
-------------------------------------------------------------------
end HANDLE_MENU_CHOICE;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_MOUSE_DOWN is
-------------------------------------------------------------------
  THIS_WINDOW     : QUICKDRAW.WINDOWPTR;
  WINDOW_SIZE     : MAC_TYPES.LONGINT;
  OLD_PORT        : QUICKDRAW.GRAFPTR;
  NORMAL_UPDATES  : BOOLEAN := TRUE;
  THE_POINT       : MAC_TYPES.POINT;
  THE_PART        : MAC_TYPES.INTEGER;
  THE_MENU        : MAC_TYPES.LONGINT;
-------------------------------------------------------------------
begin

  THE_PART := WINDOWS.FINDWINDOW
              (GEVENT.WHERE,
               WINDOW_PACKAGE.
               ADDRESS_TO_VARWINDOWPTR
               (THIS_WINDOW'ADDRESS));

  case THE_PART is
    when WINDOWS.INSYSWINDOW => null;
    when WINDOWS.INMENUBAR   => THE_MENU := MENUS.
                                            MENUSELECT
                                            (STARTPT => GEVENT.
                                                        WHERE);

                                -- Select appropriate action

                                HANDLE_MENU_CHOICE
                                (SELECTED => THE_MENU);
                                MENUS.HILITEMENU(0);
```

```
    when WINDOWS.INDRAG       => WINDOWS.
                                 DRAGWINDOW
                                 (THEWINDOW  => THIS_WINDOW,
                                  STARTPT    => GEVENT.WHERE,
                                  BOUNDSRECT => QUICKDRAW.
                                                QD.
                                                SCREENBITS.
                                                BOUNDS);
    when WINDOWS.INCONTENT    => if THIS_WINDOW /=
                                    WINDOWS.FRONTWINDOW then

                                   WINDOWS.
                                   SELECTWINDOW
                                   (THEWINDOW => THIS_WINDOW);
                                 end if;
    when WINDOWS.INGOAWAY     => WINDOWS.DISPOSEWINDOW(GWINDOW);
                                 MENUS.CLEARMENUBAR;
                                 MENUS.DRAWMENUBAR;
                                 MENUS.SETMENUBAR
                                        (MENULIST => GMENU_BAR);
                                 MENUS.DRAWMENUBAR;
                                 GDONE := TRUE;
    when WINDOWS.INZOOMIN     => null;
    when WINDOWS.INZOOMOUT    => null;
    when others               => null;
  end case;
-------------------------------------------------------------------
end HANDLE_MOUSE_DOWN;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_UPDATE_EVENT is
-------------------------------------------------------------------
begin
  if LONGINT_TO_WINDOWPTR(GEVENT.MESSAGE) = GWINDOW then
    WINDOWS.BEGINUPDATE(LONGINT_TO_WINDOWPTR(GEVENT.MESSAGE));
    DRAW_GRID;
    DRAW_TEXT(GDEGREE);
    WINDOWS.ENDUPDATE(LONGINT_TO_WINDOWPTR(GEVENT.MESSAGE));
  end if;
-------------------------------------------------------------------
end HANDLE_UPDATE_EVENT;
-------------------------------------------------------------------


-------------------------------------------------------------------
procedure HANDLE_EVENT is
-------------------------------------------------------------------
  GOT_EVENT    : BOOLEAN;
  SLEEP        : constant := 60;
-------------------------------------------------------------------
begin

  if GWNEINPLEMENTED then
    GOT_EVENT := EVENTS.
                 WAITNEXTEVENT
                 (EVENTS.EVERYEVENT, GEVENT, SLEEP, null);
  else
    DESK.SYSTEMTASK;
```

```
      GOT_EVENT := EVENTS.GETNEXTEVENT(EVENTS.EVERYEVENT, GEVENT);
    end if;

    if GOT_EVENT then
      case GEVENT.WHAT is
        when EVENTS.NULLEVENT   => null;
        when EVENTS.MOUSEDOWN   => HANDLE_MOUSE_DOWN;
        when EVENTS.MOUSEUP     => null;
        when EVENTS.KEYDOWN     => null;
        when EVENTS.KEYUP       => null;
        when EVENTS.AUTOKEY     => null;
        when EVENTS.UPDATEEVT   => HANDLE_UPDATE_EVENT;
        when EVENTS.ACTIVATEEVT => null;
        when others             => null;
      end case;
    end if;
  ------------------------------------------------------------------
  end HANDLE_EVENT;
  ------------------------------------------------------------------


  ------------------------------------------------------------------
  procedure DRIVER is
  ------------------------------------------------------------------
    WNE_TRAP_NUM    : constant := 16#60#;
    UNIMPL_TRAP_NUM : constant := 16#9F#;
  ------------------------------------------------------------------
  begin

    TIMETABLE_SCHEDULER;

    WINDOW_PACKAGE.INITIALISE_WINDOW;
    INITIALISE_MENU_BAR;

    QUICKDRAW.TEXTFONT(GFONT);
    QUICKDRAW.TEXTSIZE(GFONT_SIZE);

    SET_DRAWING_PARAMETERS;

    DRAW_GRID;
    DRAW_TEXT(GDEGREE);

    GWNEINPLEMENTED := OSUTILS.
                       NGETTRAPADDRESS
                       (WNE_TRAP_NUM, OSUTILS.TOOLTRAP) /=
                       OSUTILS.
                       NGETTRAPADDRESS
                       (UNIMPL_TRAP_NUM, OSUTILS.TOOLTRAP);

    while not GDONE
    loop
      WINDOW_PACKAGE.HANDLE_EVENT;
    end loop;
  ------------------------------------------------------------------
  end DRIVER;
  ------------------------------------------------------------------

------------------------------------------------------------------
end WINDOW_PACKAGE;
------------------------------------------------------------------
```

Timetable Knowledge

ESEp1mact.rule

```
IF   E101(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11_MON-1

IF   E101(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1,3,5,7,9,11_WED-2

IF   E101(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2,4,6,8,10_WED-2

IF   E101(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E101(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2,4,6,8,10_FRI-4

IF   E151(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E151(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_4,6,9,10

IF   E151(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3-11

IF   E151(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_3,6,9-11

IF   E170(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E170(T)
THEN allocate_1_/_3_period_Tutorial_in_weeks_1-11

IF   E170(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_1-11

IF   E102(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E102(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_1-10

IF   E102(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2-11

IF   E132(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E132(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_1-10

IF   E132(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2-11
```

```
IF   E103Design(L)
THEN allocate_2_/_2_period_Lectures_in_week_1_TUE-3_THU-3

IF   E103Mp(L)
THEN allocate_1_/_2_period_Lecture_in_weeks_7,10_TUE-3

IF   E103Mp(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_7-11

IF   E103Mp(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_7-8,10-11

IF   E103Basic(P)
THEN allocate_1_/_2_period_Practical_in_weeks_2-6,8-9,11_TUE-3

IF   E103Basic(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_1-10

IF   E103Basic(P)
THEN allocate_1_/_2_period_Practical_in_weeks_2-5,7-10_THU-3

IF   E133(C)
THEN allocate_2_/_3_period_Carousel_in_weeks_1-11

IF   E152(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_3,5,7,11

IF   E152(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3-11

IF   E152(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E152(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_2,4,7-8

IF   E153(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_3-4,8-9

IF   E153(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3-11

IF   E153(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E153(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_2,5-7,9,11

IF   E131(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   E131(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2-11

IF   E131(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_1-10
```

ESEp1mcom.rule

```
IF   E101(L) AND I111(L)
THEN make_E101_and_I111_common

IF   E101(L) AND Q111(L)
THEN make_E101_and_Q111_common

IF   E151(L) AND I119(L)
THEN make_E151(L)_and_I119(L)_common

IF   E151(C) AND I119(C)
THEN make_E151(C)_and_I119(C)_common

IF   E151(L) AND Q119(L)
THEN make_E151(L)_and_Q119(L)_common

IF   E151(C) AND Q119(C)
THEN make_E151(C)_and_Q119(C)_common

IF   E151(T) AND E152(T)
THEN make_E151(T)_and_E152(T)_common

IF   E151(T) AND E153(T)
THEN make_E151(T)_and_E153(T)_common

IF   E102(T) AND E131(T)
THEN make_E102(T)_and_E131(T)_common

IF   E102(T) AND E132(T)
THEN make_E102(T)_and_E132(T)_common

IF   E132(T) AND E131(T)
THEN make_E132(T)_and_E131(T)_common

IF   E132(T) AND E102(T)
THEN make_E132(T)_and_E102(T)_common

IF   E103Design(L) AND I114Design(L)
THEN make_E103Design(L)_and_I114Design(L)_common

IF   E103Design(L) AND Q114Design(L)
THEN make_E103Design(L)_and_Q114Design(L)_common

IF   E103Mp(L) AND I116Mp(L)
THEN make_E103Mp(L)_and_I116Mp(L)_common

IF   E103Mp(C) AND I116Mp(C)
THEN make_E103Mp(C)_and_I116Mp(C)_common

IF   E103Mp(L) AND Q116Mp(L)
THEN make_E103Mp(L)_and_Q116Mp(L)_common

IF   E103Mp(C) AND Q116Mp(C)
THEN make_E103Mp(C)_and_Q116Mp(C)_common

IF   E152(L) AND Q152(L)
THEN make_E152(L)_and_Q152(L)_common
```

```
IF   E152(C) AND Q152(C)
THEN make_E152(C)_and_Q152(C)_common

IF   E152(L) AND S112(L)
THEN make_E152(L)_and_S112(L)_common

IF   E152(T) AND E151(T)
THEN make_E152(T)_and_E151(T)_common

IF   E152(T) AND E153(T)
THEN make_E152(T)_and_E153(T)_common

IF   E153(L) AND I117(L)
THEN make_E153(L)_and_I117(L)_common

IF   E153(C) AND I117(C)
THEN make_E153(C)_and_I117(C)_common

IF   E153(L) AND Q117(L)
THEN make_E153(L)_and_Q117(L)_common

IF   E153(C) AND Q117(C)
THEN make_E153(C)_and_Q117(C)_common

IF   E153(T) AND E151(T)
THEN make_E153(T)_and_E151(T)_common

IF   E153(T) AND E152(T)
THEN make_E153(T)_and_E152(T)_common

IF   E131(T) AND E132(T)
THEN make_E131(T)_and_E132(T)_common

IF   E131(T) AND E102(T)
THEN make_E131(T)_and_E102(T)_common
```

ESEp1mmod.list

```
E151 50
E152 50
E153 50
E102 50
E131 50
E132 50
E133 50
E170 50
E101 50
E103 100
```

ESEp1mreq.rule

```
IF   E101
THEN provide_lecture_tutorial

IF   E151
THEN provide_lecture_tutorial_carousel
```

```
IF   E170
THEN provide_lecture_tutorial

IF   E102
THEN provide_lecture_tutorial_carousel

IF   E132
THEN provide_lecture_tutorial_carousel

IF   E103Design
THEN provide_lecture

IF   E103Mp
THEN provide_lecture_carousel

IF   E103Basic
THEN provide_practical_carousel

IF   E133
THEN provide_carousel

IF   E152
THEN provide_lecture_tutorial_carousel

IF   E153
THEN provide_lecture_tutorial_carousel

IF   E131
THEN provide_lecture_tutorial_carousel
```

ESEp1msub.rule

```
IF   E103
THEN E103Design

IF   E103
THEN E103Basic

IF   E103
THEN E103Mp
```

ESEperiod.pro

```
period(mon, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p4, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p5, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p6, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p7, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p4, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p5, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(tue, p6, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
```

```
period(tue, p7, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p1, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p2, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p3, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p4, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p5, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p6, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p7, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(thu, p1, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p2, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p3, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p4, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p5, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(thu, p6, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(thu, p7, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(fri, p1, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p2, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p3, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p4, wks(o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p5, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(fri, p6, wks(x,x,x,x,x,x,x,x,x,x,x)).
period(fri, p7, wks(x,x,x,x,x,x,x,x,x,x,x)).

one_period(D, P, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11))
:-
period(D, p1, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11)),
P is p1.

one_period(D, P, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11))
:-
period(D, p2, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11)),
P is p2.

one_period(D, P, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11))
:-
period(D, p3, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11)),
P is p3.

one_period(D, P, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11))
:-
period(D, p4, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11)),
P is p4.

two_periods(D, P1, P2,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11))
:-
period(D,p1,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
P1W11)),
period(D,p2,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
P2W11)),
P1 is p1,
P2 is p2.

two_periods(D, P1, P2,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11))
:-
period(D,p3,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
P1W11)),
```

```
period(D,p4,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
P2W11)),
P1 is p3,
P2 is p4.

three_periods(D, P1, P2, P3,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11),
wks(P3W1,P3W2,P3W3,P3W4,P3W5,P3W6,P3W7,P3W8,P3W9,P3W10,P3W11))
:-
period(D,p5,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
P1W11)),
period(D,p6,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
P2W11)),
period(D,p7,wks(P3W1,P3W2,P3W3,P3W4,P3W5,P3W6,P3W7,P3W8,P3W9,P3W10,
P3W11)),
P1 is p5,
P2 is p6,
P3 is p7.
```

ITp1mact.rule

```
IF   I111(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I111(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1,3,5,7,9,11

IF   I111(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2,4,6,8,10

IF   I111(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I119(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I119(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3,6,8-10

IF   I119(C)
THEN allocate_1_/_1_period_Carousel_in_weeks_3,9

IF   I113(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I113(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3,5,7,9

IF   I117(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_4,7,9,11

IF   I117(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I117(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_7,11
```

```
IF   I118(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2,4,7,10

IF   I118(L)
THEN allocate_1_/_2_period_Lecture_in_weeks_1-4,6-8,10

IF   I123(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3,5,8,10

IF   I123(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I123(P)
THEN allocate_1_/_3_period_Practical_in_week_10

IF   I114Design(L)
THEN allocate_2_/_2_period_Lectures_in_week_1

IF   I114Prolog(T)
THEN allocate_1_/_1_period_Lecture_in_weeks_2-11

IF   I114Prolog(P)
THEN allocate_1_/_1_period_Practical_in_weeks_5,7,9,11

IF   I116Tools(P)
THEN allocate_1_/_1_period_Practical_in_weeks_2-6,8,9,11

IF   I116Tools(P)
THEN allocate_1_/_1_period_Practical_in_weeks_3-5,8,9,11

IF   I116Mp(L)
THEN allocate_1_/_2_period_Lecture_in_weeks_7,10

IF   I116Mp(C)
THEN allocate_1_/_3_period_Carousel_in_weeks_9,11

IF   I122(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I122(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_2,4,7,10

IF   I125(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I115(L)
THEN allocate_1_/_2_period_Lecture_in_weeks_5,9

IF   I115(L)
THEN allocate_1_/_1_period_Lecture_in_weeks_1-11

IF   I115(T)
THEN allocate_1_/_1_period_Tutorial_in_weeks_3,6,8,10
```

ITp1mcom.rule

```
IF   I111(L) AND Q111(L)
THEN make_I111(L)_and_Q111(L)_common
```

```
IF   I111(L) AND E101(L)
THEN make_I111(L)_and_E101(L)_common

IF   I111(T) AND Q111(T)
THEN make_I111(T)_and_Q111(T)_common

IF   I119(L) AND E151(L)
THEN make_I119(L)_and_E151(L)_common

IF   I119(C) AND E151(C)
THEN make_I119(C)_and_E151(C)_common

IF   I119(T) AND Q119(T)
THEN make_I111(T)_and_Q119(T)_common

IF   I119(C) AND Q119(C)
THEN make_I119(C)_and_Q119(C)_common

IF   I113(L) AND Q113(L)
THEN make_I113(L)_and_Q113(L)_common

IF   I113(T) AND Q113(T)
THEN make_I113(T)_and_Q113(T)_common

IF   I113(L) AND C105(L)
THEN make_I113(L)_and_C105(L)_common

IF   I113(T) AND C105(T)
THEN make_I113(T)_and_C105(T)_common

IF   I117(L) AND Q117(L)
THEN make_I117(L)_and_Q117(L)_common

IF   I117(L) AND E153(L)
THEN make_I117(L)_and_E153(L)_common

IF   I117(T) AND Q117(T)
THEN make_I117(T)_and_Q117(T)_common

IF   I117(C) AND Q117(C)
THEN make_I117(C)_and_Q117(C)_common

IF   I117(C) AND E153(C)
THEN make_I117(C)_and_E153(C)_common

IF   I118(L) AND Q118(L)
THEN make_I118(L)_and_Q118(L)_common

IF   I118(T) AND Q118(T)
THEN make_I118(T)_and_Q118(T)_common

IF   I118(L) AND C103(L)
THEN make_I118(L)_and_C103(L)_common

IF   I118(T) AND C103(T)
THEN make_I118(T)_and_C103(T)_common

IF   I123(L) AND Q123(L)
THEN make_I123(L)_and_Q123(L)_common
```

```
IF   I123(T) AND Q123(T)
THEN make_I123(T)_and_Q123(T)_common

IF   I123(P) AND Q123(P)
THEN make_I123(P)_and_Q123(P)_common

IF   I114Design(L) AND Q114Design(L)
THEN make_I114Design(L)_and_Q114Design(L)_common

IF   I114Design(L) AND E103Design(L)
THEN make_I114Design(L)_and_E103Design(L)_common

IF   I114Prolog(P) AND Q114Prolog(P)
THEN make_I114Prolog(P)_and_Q114Prolog(P)_common

IF   I114Prolog(T) AND Q114Prolog(T)
THEN make_I114Proog(T)_and_Q114Prolog(T)_common

IF   I116Tools(P) AND Q116Tools(P)
THEN make_I116Tools(P)_and_Q116Tools(P)_common

IF   I116Mp(L) AND E103Mp(L)
THEN make_I116Mp(L)_and_E103Mp(L)_common

IF   I116Mp(C) AND E103Mp(C)
THEN make_I116Mp(C)_and_E103Mp(C)_common

IF   I116Mp(L) AND Q116Mp(L)
THEN make_I116Mp(L)_and_Q116Mp(L)_common

IF   I116Mp(C) AND Q116Mp(C)
THEN make_I116Mp(C)_and_Q116Mp(C)_common

IF   I122(L) AND Q122(L)
THEN make_I122(L)_and_Q112(L)_common

IF   I122(T) AND Q122(T)
THEN make_I122(T)_and_Q122(T)_common

IF   I122(L) AND C110(L)
THEN make_I122(L)_and_C110(L)_common

IF   I122(T) AND C110(T)
THEN make_I122(T)_and_C110(T)_common

IF   I125(L) AND Q125(L)
THEN make_I125(L)_and_Q125(L)_common

IF   I122(L) AND C113(L)
THEN make_I122(L)_and_C113(L)_common

IF   I115(L) AND Q115(L)
THEN make_I115(L)_and_Q115(L)_common

IF   I115(L) AND C108(L)
THEN make_I115(L)_and_C108(L)_common

IF   I115(T) AND Q115(T)
THEN make_I115(T)_and_Q115(T)_common
```

```
IF   I115(T) AND C108(T)
THEN make_I115(T)_and_C108(T)_common
```

ITp1mmod.list

```
I111 20
I113 20
I114 20
I115 20
I116 20
I117 20
I118 20
I119 20
I122 20
I123 20
I125 20
```

ITp1mreq.rule

```
IF   I111
THEN provide_lecture_tutorial

IF   I119
THEN provide_lecture_tutorial_carousel

IF   I113
THEN provide_lecture_tutorial

IF   I117
THEN provide_lecture_tutorial_carousel

IF   I118
THEN provide_lecture_tutorial

IF   I123
THEN provide_lecture_tutorial_practical

IF   I114Design
THEN provide_lecture

IF   I114Prolog
THEN provide_tutorial_practical

IF   I116Tools
THEN provide_practical

IF   I116Mp
THEN provide_lecture_carousel

IF   I122
THEN provide_lecture_tutorial

IF   I125
THEN provide_lecture
```

```
IF   I115
THEN provide_lecture_tutorial
```

ITp1msub.rule

```
IF   I114
THEN I114Design

IF   I114
THEN I114Prolog

IF   I116
THEN I116Tools

IF   I116
THEN I116Mp
```

ITperiod.pro

```
period(mon, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p4, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p5, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p6, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(mon, p7, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p4, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p5, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p6, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(tue, p7, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(wed, p4, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p5, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p6, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(wed, p7, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(thu, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p4, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p5, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p6, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(thu, p7, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p1, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p2, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p3, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p4, wks(o,o,o,o,o,o,o,o,o,o,o,o)).
period(fri, p5, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(fri, p6, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
period(fri, p7, wks(x,x,x,x,x,x,x,x,x,x,x,x)).
```

```
one_period(D, P, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11))
:-
period(D, P, wks(BW1,BW2,BW3,BW4,BW5,BW6,BW7,BW8,BW9,BW10,BW11)).


two_periods(D, P1, P2,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11))
:-
period(D,p1,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
P1W11)),
period(D,p2,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
P2W11)),
P1 is p1,
P2 is p2.

two_periods(D, P1, P2,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11))
:-
period(D,p3,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
P1W11)),
period(D,p4,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
P2W11)),
P1 is p3,
P2 is p4.

three_periods(D, P1, P2, P3,
wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,P1W11),
wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,P2W11),
wks(P3W1,P3W2,P3W3,P3W4,P3W5,P3W6,P3W7,P3W8,P3W9,P3W10,P3W11))
:-
period(D,p5,wks(P1W1,P1W2,P1W3,P1W4,P1W5,P1W6,P1W7,P1W8,P1W9,P1W10,
P1W11)),
period(D,p6,wks(P2W1,P2W2,P2W3,P2W4,P2W5,P2W6,P2W7,P2W8,P2W9,P2W10,
P2W11)),
period(D,p7,wks(P3W1,P3W2,P3W3,P3W4,P3W5,P3W6,P3W7,P3W8,P3W9,P3W10,
P3W11)),
P1 is p5,
P2 is p6,
P3 is p7.
```

p1mactfile.list

```
ESEp1mact.rule

ITp1mact.rule
```

p1mcomfile.list

```
ESEp1mcom.rule

ITp1mcom.rule
```

p1mmodfile.list

```
ESEp1mmod.list
ITp1mmod.list
```

p1mreqfile.list

```
ESEp1mreq.rule
ITp1mreq.rule
```

p1msubfile.list

```
ESEp1msub.rule
ITp1msub.rule
```

room.frame

```
CLASS ROOM SUPERCLASS ROOT

CLASS LT SUPERCLASS ROOM

CLASS LLR SUPERCLASS ROOM

CLASS SLR SUPERCLASS ROOM

CLASS TUT SUPERCLASS ROOM

CLASS ELAB SUPERCLASS ROOM

CLASS CLAB SUPERCLASS ROOM

INSTANCE LLT SUPERCLASS LT SLOT DETAILS FACET CAPACITY 144

INSTANCE LFLT SUPERCLASS LT SLOT DETAILS FACET CAPACITY 240

INSTANCE RLT SUPERCLASS LT SLOT DETAILS FACET CAPACITY 112

INSTANCE WH42 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 79

INSTANCE WHLT SUPERCLASS LT SLOT DETAILS FACET CAPACITY 206

INSTANCE WH190 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 28

INSTANCE WH 196 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 28

INSTANCE WH241 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 30

INSTANCE WH250 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 20

INSTANCE RDH SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 30

INSTANCE WH103 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 38

INSTANCE WH 208 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 12

INSTANCE WH248 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 12

INSTANCE WH276 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 30

INSTANCE MH1 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 72

INSTANCE MH7 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 20

INSTANCE MH69 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 16

INSTANCE MH88 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 30

INSTANCE MH125B SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 25

INSTANCE MH127B SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 24

INSTANCE SL6B SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 38

INSTANCE SL6A SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 24
```

```
INSTANCE MH169 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 72
INSTANCE MH221 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 72
INSTANCE MH223 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 42
INSTANCE MH121 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 40
INSTANCE MH269 SUPERCLASS SLR SLOT DETAILS FACET CAPACITY 42
INSTANCE MH122 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 12
INSTANCE MH148 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 14
INSTANCE MH92 SUPERCLASS LLR SLOT DETAILS FACET CAPACITY 25
INSTANCE MH112 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 12
INSTANCE MH117 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 14
INSTANCE MH125A SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 14
INSTANCE MH126 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 14
INSTANCE MH270 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 10
INSTANCE SL1 SUPERCLASS TUT SLOT DETAILS FACET CAPACITY 18
```

room.pro

```
room_periods_1
(R1,S1,D1,P1,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11))
:-
room
(R1,C1,D1,P1,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11)),
S1 < C1.

room_periods_2
(R1,S1,D1,P1,P2,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11))
:-
room
(R1,C1,D1,P1,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11)),
S1 < C1,
room
(R1,C1,D1,P2,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11)).

room_periods_3
(R1,S1,D1,P1,P2,P3,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11))
:-
room
(R1,C1,D1,P1,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11)),
S1 < C1,
room(R1,C1,D1,P2,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11)),
room(R1,C1,D1,P3,w(W1,W2,W3,W4,W5,W6,W7,W8,W9,W10,W11)).

mh121(40).
mh223(42).
mh269(42).
mh169(72).
mh221(72).
llt(144).
lflt(240).

room(mh121,  40, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, mon, p2, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, mon, p3, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, mon, p4, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, mon, p5, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, mon, p6, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, mon, p7, w(o,o,o,o,o,o,o,o,o,o,o)).

room(mh121,  40, tue, p1, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, tue, p2, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, tue, p3, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, tue, p4, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, tue, p5, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, tue, p6, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, tue, p7, w(o,o,o,o,o,o,o,o,o,o,o)).

room(mh121,  40, wed, p1, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, wed, p2, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, wed, p3, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, wed, p4, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, wed, p5, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, wed, p6, w(o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, wed, p7, w(o,o,o,o,o,o,o,o,o,o,o)).
```

```
room(mh121,  40, thu, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, thu, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, thu, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, thu, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, thu, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, thu, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, thu, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh121,  40, fri, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, fri, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, fri, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, fri, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, fri, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, fri, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh121,  40, fri, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh223,  42, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, mon, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, mon, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, mon, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, mon, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, mon, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, mon, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh223,  42, tue, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, tue, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, tue, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, tue, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, tue, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, tue, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, tue, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh223,  42, wed, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, wed, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, wed, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, wed, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, wed, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, wed, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, wed, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh223,  42, thu, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, thu, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, thu, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, thu, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, thu, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, thu, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, thu, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh223,  42, fri, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, fri, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, fri, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, fri, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, fri, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, fri, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh223,  42, fri, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh269,  42, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, mon, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, mon, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, mon, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
```

```
room(mh269,  42, mon, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, mon, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, mon, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh269,  42, tue, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, tue, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, tue, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, tue, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, tue, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, tue, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, tue, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh269,  42, wed, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, wed, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, wed, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, wed, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, wed, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, wed, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, wed, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh269,  42, thu, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, thu, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, thu, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, thu, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, thu, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, thu, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, thu, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh269,  42, fri, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, fri, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, fri, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, fri, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, fri, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, fri, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh269,  42, fri, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh169,  72, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, mon, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, mon, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, mon, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, mon, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, mon, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, mon, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh169,  72, tue, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, tue, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, tue, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, tue, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, tue, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, tue, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, tue, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh169,  72, wed, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, wed, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, wed, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, wed, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, wed, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, wed, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,  72, wed, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).
```

```
room(mh169,   72, thu, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, thu, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, thu, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, thu, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, thu, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, thu, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, thu, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(mh169,   72, fri, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, fri, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, fri, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, fri, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, fri, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, fri, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(mh169,   72, fri, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(llt,    144, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, mon, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, mon, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, mon, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, mon, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, mon, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, mon, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(llt,    144, tue, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, tue, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, tue, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, tue, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, tue, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, tue, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, tue, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(llt,    144, wed, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, wed, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, wed, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, wed, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, wed, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, wed, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, wed, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(llt,    144, thu, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, thu, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, thu, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, thu, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, thu, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, thu, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, thu, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(llt,    144, fri, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, fri, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, fri, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, fri, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, fri, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, fri, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(llt,    144, fri, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(lflt,   240, mon, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,   240, mon, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,   240, mon, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,   240, mon, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
```

```
room(lflt,  240, mon, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, mon, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, mon, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(lflt,  240, tue, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, tue, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, tue, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, tue, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, tue, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, tue, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, tue, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(lflt,  240, wed, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, wed, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, wed, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, wed, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, wed, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, wed, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, wed, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(lflt,  240, thu, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, thu, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, thu, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, thu, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, thu, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, thu, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, thu, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).

room(lflt,  240, fri, p1, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, fri, p2, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, fri, p3, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, fri, p4, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, fri, p5, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, fri, p6, w(o,o,o,o,o,o,o,o,o,o,o,o)).
room(lflt,  240, fri, p7, w(o,o,o,o,o,o,o,o,o,o,o,o)).
```

staffmoduleESE.frame

```
CLASS ESE SUPERCLASS ROOT

CLASS Part1_ESE SUPERCLASS ESE

CLASS Part2_ESE SUPERCLASS ESE

CLASS Part3_ESE SUPERCLASS ESE

INSTANCE E101(L) SUPERCLASS Part1_ESE SLOT Staff FACET DCS Stocks

INSTANCEE101(T) SUPERCLASS Part1_ESE SLOT Staff FACET DCS Stocks

INSTANCE E151(L) SUPERCLASS Part1_ESE SLOT Staff FACET WGT Townsend

INSTANCE E151(C) SUPERCLASS Part1_ESE SLOT Staff FACET AWB Bingham
FACET DJD Diskett

INSTANCE E151(T) SUPERCLASSPart1_ESE SLOT Staff FACET WGT Townsend
FACET PMGS Silson

INSTANCE E170(L) SUPERCLASS Part1_ESE SLOT Staff FACET DEE Eldred
FACET KRM McNaught

INSTANCE E170(T) SUPERCLASS Part1_ESE SLOT Staff FACET DEE Eldred
FACET KRM MaNaught

INSTANCE E102(L) SUPERCLASS Part1_ESE SLOT Staff FACET RDB Brown

INSTANCE E102(C) SUPERCLASS Part1_ESE SLOT Staff FACET RDB Brown

INSTANCE E102(T) SUPERCLASS Part1_ESE SLOT Staff FACET RDB Brown

INSTANCE E132(L) SUPERCLASS Part1_ESE SLOT Staff FACET PJHW Wormell

INSTANCE E132(C) SUPERCLASS Part1_ESE SLOT Staff FACET PJHW Wormell

INSTANCE E132(T) SUPERCLASS Part1_ESE SLOT Staff FACET PJHW Wormell

INSTANCE E103Design(L) SUPERCLASS Part1_ESE SLOT Staff FACET AH
Harrison

INSTANCE E103Mp(L) SUPERCLASS Part1_ESE SLOT Staff FACET LFJ Jardine

INSTANCE E103Mp(C) SUPERCLASS Part1_ESE SLOT Staff FACET LFJ Jardine

INSTANCE E103Basic(P) SUPERCLASS Part1_ESE SLOT Staff FACET BF
Farmillo

INSTANCE E103Basic(C) SUPERCLASS Part1_ESE SLOT Staff FACET BF
Farmillo

INSTANCE E133(C) SUPERCLASS Part1_ESE SLOT Staff FACET RW Whitford

INSTANCE E152(C) SUPERCLASS Part1_ESE SLOT Staff FACET BLJ Jones

INSTANCE E152(T) SUPERCLASS Part1_ESE SLOT Staff FACET BLJ Jones
FACET REC Colyer
```

```
INSTANCE E152(L) SUPERCLASS Part1_ESE SLOT Staff FACET BLJ Jones

INSTANCE E153(C) SUPERCLASS Part1_ESE SLOT Staff FACET HGB Brierley

INSTANCE E153(T) SUPERCLASS Part1_ESE SLOT Staff FACET HGB Brierley
FACET JEA Aitken

INSTANCE E153(L) SUPERCLASS Part1_ESE SLOT Staff FACET HGB Brierley

INSTANCE E131(L) SUPERCLASS Part1_ESE SLOT Staff FACET PJM Moss

INSTANCE E131(T) SUPERCLASS Part1_ESE SLOT Staff FACET PJM Moss

INSTANCE E131(C) SUPERCLASS Part1_ESE SLOT Staff FACET PJM Moss
```

staffmoduleIT.frame

```
CLASS IT SUPERCLASSROOT

CLASS Part1_IT SUPERCLASS IT

CLASS Part2_IT SUPERCLASS IT

CLASS Part3_IT SUPERCLASS IT

INSTANCE I111(L) SUPERCLASS Part1_IT SLOT Staff FACET DCS Stocks

INSTANCE I111(T) SUPERCLASS Part1_IT SLOT Staff FACET DCS Stocks

INSTANCE I119(L) SUPERCLASS Part1_IT SLOT Staff FACET WGT Towsend

INSTANCE I119(T) SUPERCLASS Part1_IT SLOT Staff FACET WGT Townsend

INSTANCE I119(C) SUPERCLASS Part1_IT SLOT Staff FACET AWB Bingham
FACET DJD Diskett

INSTANCE I113(L) SUPERCLASS Part1_IT SLOT Staff FACET BJH Hilton

INSTANCE I113(T) SUPERCLASS Part1_IT SLOT Staff FACET BJH Hilton

INSTANCE I117(L) SUPERCLASS Part1_IT SLOT Staff FACET HGB Brierley

INSTANCE I117(T) SUPERCLASS Part1_IT SLOT Staff FACET HGB Brierley
FACET JEA Aitken

INSTANCE I117(C) SUPERCLASS Part1_IT SLOT Staff FACET HGB Brierley

INSTANCE I118(L) SUPERCLASS Part1_IT SLOT Staff FACET JCM Mason

INSTANCE I118(T) SUPERCLASS Part1_IT SLOT Staff FACET JCM Mason

INSTANCE I123(L) SUPERCLASS Part1_IT SLOT Staff FACET JDP Pryce

INSTANCE I123(T) SUPERCLASS Part1_IT SLOT Staff FACET JDP Pryce

INSTANCE I123(P) SUPERCLASS Part1_IT SLOT Staff FACET JDP Pryce

INSTANCE I114Design(L) SUPERCLASS Part1_IT SLOT Staff FACET AH
```

```
Harrison

INSTANCE I114Prolog(L) SUPERCLASS Part1_IT SLOT Staff FACET JDP
Pryce

INSTANCE I114Prolog(T) SUPERCLASS Part1_IT SLOT Staff FACET JDP
Pryce

INSTANCE I114Prolog(P) SUPERCLASS Part1_IT SLOT Staff FACET JDP
Pryce

INSTANCE I116Tools(P) SUPERCLASS Part1_IT SLOT Staff FACET WHTK King

INSTANCE I116Mp(L) SUPERCLASS Part1_IT SLOT Staff FACET LFJ Jardine

INSTANCE I116Mp(C) SUPERCLASS Part1_IT SLOT Staff FACET LFJ Jardine

INSTANCE I122(L) SUPERCLASS Part1_IT SLOT Staff FACET LW Watson

INSTANCE I122(T) SUPERCLASS Part1_IT SLOT Staff FACET LW Watson

INSTANCE I125(L) SUPERCLASS Part1_IT SLOT Staff FACET DEE Eldred

INSTANCE I115(L) SUPERCLASS Part1_IT SLOT Staff FACET MLV Vaughn

INSTANCE I115(T) SUPERCLASS Part1_IT SLOT Staff FACET MLV Vaughn
```

staffSDM.frame

```
CLASS SDM SUPERCLASS ROOT

CLASS ACM SUPERCLASS SDM

CLASS CISM SUPERCLASS SDM

CLASS EPSS SUPERCLASS SDM

CLASS MS SUPERCLASS SDM

CLASS MSystems SUPERCLASS SDM

CLASS SA SUPERCLASS SDM

INSTANCE RAM SUPERCLASS SDM SLOT DETAILS FACET Miller SLOT SUBJECTS
FACET ALL

INSTANCE DCS SUPERCLASS ACM SLOT DETAILS FACET Stocks SLOT SUBJECTS
FACET ALL FACET I111(L) FACET Q111(L) FACET E101(L) FACET I111(T)
FACET Q111(T) FACET E101(T)

INSTANCE JCM SUPERCLASS ACM SLOT DETAILS FACET Mason SLOT SUBJECTS
FACET ALL FACET I118(T) FACET I118(L) FACET Q118(T) FACET Q118(L)
FACET C103(L) FACET C103(T) FACET I115(L) FACET Q115(L) FACET
C108(L) FACET I115(T) FACET Q115(T) FACET C108(T)

INSTANCE PCP SUPERCLASS ACM SLOT DETAILS FACET Parks SLOT SUBJECTS
FACET ALL INSTANCE ABC SUPERCLASS ACM SLOT DETAILS FACET Crowley
SLOT SUBJECTS FACET ALL

INSTANCE JMA SUPERCLASS ACM SLOT DETAILS FACET Aitchison SLOT
SUBJECTS FACET ALL INSTANCE RNLS SUPERCLASS ACM SLOT DETAILS FACET
Smith SLOT SUBJECTS FACET ALL

INSTANCE RPB SUPERCLASS ACM SLOT DETAILS FACET Bennell SLOT SUBJECTS
FACET ALL

INSTANCE KCJ SUPERCLASS ACM SLOT DETAILS FACET Jalie SLOT SUBJECTS
FACET ALL

INSTANCE VSS SUPERCLASS ACM SLOT DETAILS FACET Sastry SLOT SUBJECTS
FACET ALL

INSTANCE AJS SUPERCLASS CISM SLOT DETAILS FACET Sammes SLOT SUBJECTS
FACET ALL

INSTANCE MGE SUPERCLASS CISM SLOT DETAILS FACET Edwards SLOT
SUBJECTS FACET ALL

INSTANCE JMDH SUPERCLASS CISM SLOT DETAILS FACET Hunter SLOT
SUBJECTS FACET ALL

INSTANCE MPL SUPERCLASS CISM SLOT DETAILS FACET Lee SLOT SUBJECTS
FACET ALL

INSTANCE LW SUPERCLASS CISM SLOT DETAILS FACET Watson SLOT SUBJECTS
FACET ALL FACET I122(L) FACET Q122(L) FACET I122(T) FACET Q122(T)
FACET C110(L) FACET C110(T)
```

```
INSTANCE MPG SUPERCLASS CISM SLOT DETAILS FACET Griffiths SLOT
SUBJECTS FACET ALL

INSTANCE AEK SUPERCLASS CISM SLOT DETAILS FACET Kent SLOT SUBJECTS
FACET ALL

INSTANCE SR SUPERCLASS CISM SLOT DETAILS FACET Robertson SLOT
SUBJECTS FACET ALL

INSTANCE CMS SUPERCLASS CISM SLOT DETAILS FACET Streatfield SLOT
SUBJECTS FACET ALL

INSTANCE JWT SUPERCLASS CISM SLOT DETAILS FACET Thorn SLOT SUBJECTS
FACET ALL

INSTANCE MLV SUPERCLASS CISM SLOT DETAILS FACET Vaughn SLOT SUBJECTS
FACET ALL FACET I115(L) FACET Q115(L) FACET C108(L) FACET I115(T)
FACET Q115(T) FACET C108(T)

INSTANCE GEG SUPERCLASS EPSS SLOT DETAILS FACET Gibbons SLOT
SUBJECTS FACET ALL

INSTANCE RAB SUPERCLASS EPSS SLOT DETAILS FACET Bartell SLOT
SUBJECTS FACET ALL

INSTANCE DEE SUPERCLASS EPSS SLOT DETAILS FACET Eldred SLOT SUBJECTS
FACET ALL FACET I125(L) FACET Q125(L) FACET C113(L) FACET E170(L)
FACET E170(T)

INSTANCE RGM SUPERCLASS EPSS SLOT DETAILS FACET Mathews SLOT
SUBJECTS FACET ALL

INSTANCE TC SUPERCLASS MS SLOT DETAILS FACET Cass SLOT SUBJECTS
FACET ALL

INSTANCE JLH SUPERCLASS MS SLOT DETAILS FACET Halsall SLOT SUBJECTS
FACET ALL

INSTANCE SZ SUPERCLASS MS SLOT DETAILS FACET Zvegintzov SLOT
SUBJECTS FACET ALL

INSTANCE JMB SUPERCLASS MS SLOT DETAILS FACET Baskerville SLOT
SUBJECTS FACET ALL

INSTANCE WJD SUPERCLASS MS SLOT DETAILS FACET Dunn SLOT SUBJECTS
FACET ALL

INSTANCE DJE SUPERCLASS MS SLOT DETAILS FACET Edwards SLOT SUBJECTS
FACET ALL

INSTANCE MCG SUPERCLASS MS SLOT DETAILS FACET Glen SLOT SUBJECTS
FACET ALL

INSTANCE KWL SUPERCLASS MS SLOT DETAILS FACET Lambert SLOT SUBJECTS
FACET ALL

INSTANCE DHT SUPERCLASS MS SLOT DETAILS FACET Taylor SLOT SUBJECTS
FACET ALL
```

```
INSTANCE PCJH SUPERCLASS CISE SLOT DETAILS FACET Hill SLOT SUBJECTS
FACET ALL

INSTANCE HGB SUPERCLASS CISE SLOT DETAILS FACET Brierley SLOT
SUBJECTS FACET ALL FACET I117(T) FACET Q117(T) FACET I117(L) FACET
Q117(L) FACET E153(L) FACET E153(T) FACET E153(C) FACET I117(C)
FACET Q117(C)

INSTANCE IRW SUPERCLASS CISE SLOT DETAILS FACET Whitworth SLOT
SUBJECTS FACET ALL

INSTANCE JELH SUPERCLASS CISE SLOT DETAILS FACET Holis SLOT SUBJECTS
FACET ALL

INSTANCE ERA SUPERCLASS CISE SLOT DETAILS FACET Adams SLOT SUBJECTS
FACET ALL

INSTANCE VEC SUPERCLASS CISE SLOT DETAILS FACET Comeley SLOT
SUBJECTS FACET ALL

INSTANCE ML SUPERCLASS CISE SLOT DETAILS FACET Leyland SLOT SUBJECTS
FACET ALL

INSTANCE LFJ SUPERCLASS CISE SLOT DETAILS FACET Jardine SLOT
SUBJECTS FACET ALL FACET I116Mp(L) FACET Q116Mp(L) FACET I116Mp(C)
FACET Q116Mp(C) FACET E103Mp(L) FACET E103Mp(C)

INSTANCE DJS SUPERCLASS CISE SLOT DETAILS FACET Stone SLOT SUBJECTS
FACET ALL INSTANCE CWW SUPERCLASS CISE SLOT DETAILS FACET Walters
SLOT SUBJECTS FACET ALL

INSTANCE JT SUPERCLASS CISE SLOT DETAILS FACET Thickpenny SLOT
SUBJECTS FACET ALL

INSTANCE BAW SUPERCLASS CG SLOT DETAILS FACET White SLOT SUBJECTS
FACET ALL

INSTANCE JEA SUPERCLASS CG SLOT DETAILS FACET Aitken SLOT SUBJECTS
FACET ALL FACET I117(T) FACET Q117(T) FACET E153(T)

INSTANCE MGK SUPERCLASS CG SLOT DETAILS FACET Kellett SLOT SUBJECTS
FACET ALL

INSTANCE RSP SUPERCLASS CG SLOT DETAILS FACET Picton SLOT SUBJECTS
FACET ALL

INSTANCE PMGS SUPERCLASS CG SLOT DETAILS FACET Silson SLOT SUBJECTS
FACET ALL FACET E151(T)

INSTANCE AWB SUPERCLASS CG SLOT DETAILS FACET Bingham SLOT SUBJECTS
FACET ALL FACET I119(C) FACET Q119(C) FACET E151(C)

INSTANCE REC SUPERCLASS PEED SLOT DETAILS FACET Colyer SLOT SUBJECTS
FACET ALL FACET E152(T)

INSTANCE BLJ SUPERCLASS PEED SLOT DETAILS FACET Jones SLOT SUBJECTS
FACET ALL FACET E152(L) FACET Q152(L) FACET S112(L) FACET E152(T)
FACET E152(P)
```

```
INSTANCE JAEW SUPERCLASS MS SLOT DETAILS FACET Wheeler SLOT SUBJECTS
FACET ALL

INSTANCE BJH SUPERCLASS MSystems SLOT DETAILS FACET Hilton
SLOTSUBJECTS FACET ALL FACET I113(L) FACET Q113(L) FACET C105(L)
FACET I113(T) FACET Q113(T) FACET C105(T) INSTANCE SAP SUPERCLASS
MSystems SLOT DETAILS FACET Probert SLOT SUBJECTS FACET ALL

INSTANCE LE SUPERCLASS MSystems SLOT DETAILS FACET Evans SLOT
SUBJECTS FACET ALL

INSTANCE MRB SUPERCLASS SA SLOT DETAILS FACET Bathe SLOT SUBJECTS
FACET ALL

INSTANCE RGC SUPERCLASS SA SLOT DETAILS FACET Coyle SLOT SUBJECTS
FACET ALL

INSTANCE KRM SUPERCLASS SA SLOT DETAILS FACET McNaught SLOT SUBJECTS
FACET ALL FACET E170(L) FACET E170(T)
```

staffSEES.frame

```
CLASS SEES SUPERCLASS ROOT

CLASS ESE SUPERCLASS SEES

CLASS CISE SUPERCLASS SEES

CLASS CG SUPERCLASS SEES

CLASS SWENG SUPERCLASS SEES

CLASS PEED SUPERCLASS SEES

CLASS CC SUPERCLASS SEES

CLASS APEO SUPERCLASS SEES

INSTANCE WGT SUPERCLASS APEO SLOT DETAILS FACET Townsend SLOT
SUBJECTS FACET ALL FACET I119(L) FACET Q119(L) FACET E151(L) FACET
I119(T) FACET Q119(T) FACET E151(T)

INSTANCE JRJ SUPERCLASS ESE SLOT DETAILS FACET James SLOT SUBJECTS
FACET ALL

INSTANCE JSD SUPERCLASS ESE SLOT DETAILS FACET Dahele SLOT SUBJECTS
FACET ALL

INSTANCE PSH SUPERCLASS ESE SLOT DETAILS FACET Hall SLOT SUBJECTS
FACET ALL

INSTANCE RCS SUPERCLASS ESE SLOT DETAILS FACET Saull SLOT SUBJECTS
FACET ALL

INSTANCE SJV SUPERCLASS ESE SLOT DETAILS FACET Vetterlein SLOT
SUBJECTS FACET ALL
```

```
INSTANCE PRM SUPERCLASS PEED SLOT DETAILS FACET McLellan SLOT
SUBJECTS FACET ALL

INSTANCE MRM SUPERCLASS SWENG SLOT DETAILS FACET Moulding SLOT
SUBJECTS FACET ALL

INSTANCE MJH SUPERCLASS SWENG SLOT DETAILS FACET Howard SLOT
SUBJECTS FACET ALL

INSTANCE JDP SUPERCLASS SWENG SLOT DETAILS FACET Pryce SLOT SUBJECTS
FACET ALL FACET I123(L) FACET I123(T) FACET I123(P) FACET
I114Prolog(L) FACET Q114Prolog(L) FACET I114Prolog(P) FACET
Q114Prolog(P) FACET I114Prolog(T) FACET Q114Prolog(T)

INSTANCE AH SUPERCLASS SWENG SLOT DETAILS FACET Harrison SLOT
SUBJECTS FACET ALL FACET I114Design(L) FACET Q114Design(L) FACET
E103Design(L)

INSTANCE IEJ SUPERCLASS SWENG SLOT DETAILS FACET Jones SLOT SUBJECTS
FACET ALL

INSTANCE SCR SUPERCLASS SWENG SLOT DETAILS FACET Reid SLOT SUBJECTS
FACET ALL

INSTANCE BHK SUPERCLASS SWENG SLOT DETAILS FACET Hamilton-Kelly SLOT
SUBJECTS FACET ALL

INSTANCE WHTK SUPERCLASS SWENG SLOT DETAILS FACET King SLOT SUBJECTS
FACET ALL FACET I116Tools(P) FACET Q116Tools(P)

INSTANCE AN SUPERCLASS SWENG SLOT DETAILS FACET Newton SLOT SUBJECTS
FACET ALL

INSTANCE LCS SUPERCLASS SWENG SLOT DETAILS FACET Smith SLOT SUBJECTS
FACET ALL

INSTANCE DKH SUPERCLASS CC SLOT DETAILS FACET Hitchins SLOT SUBJECTS
FACET ALL

INSTANCE RFP SUPERCLASS APEO SLOT DETAILS FACET Powell SLOT SUBJECTS
FACET ALL

INSTANCE AJV SUPERCLASS APEO SLOT DETAILS FACET Avery SLOT SUBJECTS
FACET ALL

INSTANCE RHW SUPERCLASS APEO SLOT DETAILS FACET West SLOT SUBJECTS
FACET ALL

INSTANCE PWF SUPERCLASS APEO SLOT DETAILS FACET Forder SLOT SUBJECTS
FACET ALL

INSTANCE DWL SUPERCLASS APEO SLOT DETAILS FACET Lane SLOT SUBJECTS
FACET ALL

INSTANCE ICL SUPERCLASS APEO SLOT DETAILS FACET Luckcraft SLOT
SUBJECTS FACET ALL

INSTANCE MAR SUPERCLASS APEO SLOT DETAILS FACET Richardson SLOT
SUBJECTS FACET ALL
```

```
INSTANCE KDR SUPERCLASS APEO SLOT DETAILS FACET Rogers SLOT SUBJECTS
FACET ALL

INSTANCE SD SUPERCLASS APEO SLOT DETAILS FACET Dowling SLOT SUBJECTS
FACET ALL

INSTANCE SRA SUPERCLASS APEO SLOT DETAILS FACET Ahmed SLOT SUBJECTS
FACET ALL

INSTANCE DJD SUPERCLASS APEO SLOT DETAILS FACET Diskett SLOT
SUBJECTS FACET ALL FACET I119(C) FACET Q119(C) FACET E151(C)

INSTANCE KVL SUPERCLASS APEO SLOT DETAILS FACET Lovell SLOT SUBJECTS
FACET ALL

INSTANCE BJR SUPERCLASS APEO SLOT DETAILS FACET Ringgrose SLOT
SUBJECTS FACET ALL
```

staffSMMCE.frame

```
CLASS SMMCE SUPERCLASS ROOT

CLASS AS SUPERCLASS SMMCE

CLASS CS SUPERCLASS SMMCE

CLASS ASETU SUPERCLASS SMMCE

CLASS CE SUPERCLASS SMMCE

CLASS DG SUPERCLASS SMMCE

CLASS LS SUPERCLASS SMMCE

CLASS WSATSU SUPERCLASS SMMCE

CLASS MT SUPERCLASS SMMCE

CLASS TP SUPERCLASS SMMCE

CLASS WFVU SUPERCLASS SMMCE

INSTANCE DSH SUPERCLASS AS SLOT DETAILS FACET Houghton SLOT SUBJECTS
FACET ALL

INSTANCE GMM SUPERCLASS AS SLOT DETAILS FACET Moss SLOT SUBJECTS
FACET ALL

INSTANCE RW SUPERCLASS AS SLOT DETAILS FACET Whitford SLOT SUBJECTS
FACET ALL FACET E133(P)

INSTANCE DB SUPERCLASS AS SLOT DETAILS FACET Bray SLOT SUBJECTS
FACET ALL

INSTANCE KN SUPERCLASS AS SLOT DETAILS FACET Knowles SLOT SUBJECTS
FACET ALL
```

```
INSTANCE SE SUPERCLASS AS SLOT DETAILS FACET Ellis SLOT SUBJECTS
FACET ALL

INSTANCE MJS SUPERCLASS AS SLOT DETAILS FACET Simmons SLOT SUBJECTS
FACET ALL

INSTANCE ABa SUPERCLASS CS SLOT DETAILS FACET Bailey SLOT SUBJECTS
FACET ALL

INSTANCE AJB SUPERCLASS CS SLOT DETAILS FACET Bellamy SLOT SUBJECTS
FACET ALL

INSTANCE KPDC SUPERCLASS CS SLOT DETAILS FACET Clark SLOT SUBJECTS
FACET ALL

INSTANCE BD SUPERCLASS CS SLOT DETAILS FACET Dacre SLOT SUBJECTS
FACET ALL

INSTANCE JA SUPERCLASS CS SLOT DETAILS FACET Akhaven SLOT SUBJECTS
FACET ALL

INSTANCE MC SUPERCLASS CS SLOT DETAILS FACET Cartwright SLOT
SUBJECTS FACET ALL

INSTANCE DC SUPERCLASS CS SLOT DETAILS FACET Chapman SLOT SUBJECTS
FACET ALL

INSTANCE AMM SUPERCLASS CS SLOT DETAILS FACET Millinfton SLOT
SUBJECTS FACET ALL

INSTANCE SGM SUPERCLASS ASETU SLOT DETAILS FACET Murray SLOT
SUBJECTS FACET ALL

INSTANCE EGA SUPERCLASS ASETU SLOT DETAILS FACET Archer SLOT
SUBJECTS FACET ALL

INSTANCE JMB SUPERCLASS ASETU SLOT DETAILS FACET Bellerby SLOT
SUBJECTS FACET ALL

INSTANCE GCM SUPERCLASS CE SLOT DETAILS FACET Mays SLOT SUBJECTS
FACET ALL

INSTANCE WMB SUPERCLASS CE SLOT DETAILS FACET Barnes SLOT SUBJECTS
FACET ALL

INSTANCE PDS SUPERCLASS CE SLOT DETAILS FACET Smith SLOT SUBJECTS
FACET ALL

INSTANCE LJK SUPERCLASS CE SLOT DETAILS FACET Kennedy SLOT SUBJECTS
FACET ALL

INSTANCE DGR SUPERCLASS CE SLOT DETAILS FACET Rhodes SLOT SUBJECTS
FACET ALL

INSTANCE JGH SUPERCLASS DG SLOT DETAILS FACET Hetherington SLOT
SUBJECTS FACET ALL

INSTANCE MJI SUPERCLASS DG SLOT DETAILS FACET Ironmonger SLOT
SUBJECTS FACET ALL
```

```
INSTANCE RL SUPERCLASS DG SLOT DETAILS FACET Leitch SLOT SUBJECTS
FACET ALL

INSTANCE SJM SUPERCLASS DG SLOT DETAILS FACET McGuigan SLOT SUBJECTS
FACET ALL

INSTANCE PB SUPERCLASS DG SLOT DETAILS FACET Barton SLOT SUBJECTS
FACET ALL

INSTANCE RDB SUPERCLASS DG SLOT DETAILS FACET Brown SLOT SUBJECTS
FACET ALL

FACET E102(L) FACET E102(P) FACET E102(T)

INSTANCE DEA SUPERCLASS DG SLOT DETAILS FACET ELLIS SLOT SUBJECTS
FACET ALL

INSTANCE JRSU SUPERCLASS DG SLOT DETAILS FACET Uttley SLOT SUBJECTS
FACET ALL

INSTANCE DNB SUPERCLASS LS SLOT DETAILS FACET Bulman SLOT SUBJECTS
FACET ALL

INSTANCE LCH SUPERCLASS LS SLOT DETAILS FACET Hall SLOT SUBJECTS
FACET ALL

INSTANCE BEJ SUPERCLASS LS SLOT DETAILS FACET Jones SLOT SUBJECTS
FACET ALL

INSTANCE DP SUPERCLASS LS SLOT DETAILS FACET Purdy SLOT SUBJECTS
FACET ALL

INSTANCE PJHW SUPERCLASS LS SLOT DETAILS FACET Wormell SLOT SUBJECTS
FACET ALL FACET E132(L) FACET E132(T) FACET E132(P)

INSTANCE BF SUPERCLASS LS SLOT DETAILS FACET Farmilo SLOT SUBJECTS
FACET ALL FACET E103Basic(P) FACET E103Basic(C)

INSTANCE JAB SUPERCLASS MT SLOT DETAILS FACET Belk SLOT SUBJECTS
FACET ALL

INSTANCE AD SUPERCLASS MT SLOT DETAILS FACET Doig SLOT SUBJECTS
FACET ALL

INSTANCE MRE SUPERCLASS MT SLOT DETAILS FACET Edwards SLOT SUBJECTS
FACET ALL

INSTANCE CMF SUPERCLASS MT SLOT DETAILS FACET Friend SLOT SUBJECTS
FACET ALL

INSTANCE IH SUPERCLASS MT SLOT DETAILS FACET Horsfall SLOT SUBJECTS
FACET ALL

INSTANCE AB SUPERCLASS TP SLOT DETAILS FACET Brown SLOT SUBJECTS
FACET ALL

INSTANCE BL SUPERCLASS TP SLOT DETAILS FACET Lawyon SLOT SUBJECTS
FACET ALL
```

```
INSTANCE NLSF SUPERCLASS TP SLOT DETAILS FACET Filleul SLOT SUBJECTS
FACET ALL

INSTANCE PJM SUPERCLASS TP SLOT DETAILS FACET Moss SLOT SUBJECTS
FACET ALL FACET E131(L) FACET E131(T) FACET E131(P)

INSTANCE AWM SUPERCLASS TP SLOT DETAILS FACET Mowat SLOT SUBJECTS
FACET ALL

INSTANCE JDM SUPERCLASS WFVU SLOT DETAILS FACET Mackworth SLOT
SUBJECTS FACET ALL
```