ART/Ada Design Project - Phase I Task 2 Report: Detailed Design

Status Report

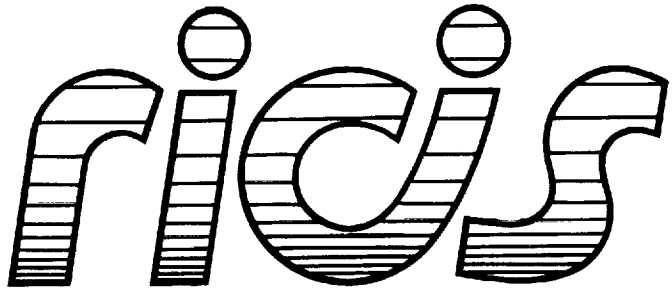
Bradley P. Allen

Inference Corporation

October 24, 1988

Cooperative Agreement NCC 9-16
Research Activity No. SE.19

NASA Johnson Space Center
Information Systems Directorate
Information Technology Division



Research Institute for Computing and Information Systems
University of Houston - Clear Lake

T·E·C·H·N·I·C·A·L R·E·P·O·R·T

The RICIS Concept

The University of Houston-Clear Lake established the Research Institute for Computing and Information systems in 1986 to encourage NASA Johnson Space Center and local industry to actively support research in the computing and information sciences. As part of this endeavor, UH-Clear Lake proposed a partnership with JSC to jointly define and manage an integrated program of research in advanced data processing technology needed for JSC's main missions, including administrative, engineering and science responsibilities. JSC agreed and entered into a three-year cooperative agreement with UH-Clear Lake beginning in May, 1986, to jointly plan and execute such research through RICIS. Additionally, under Cooperative Agreement NCC 9-16, computing and educational facilities are shared by the two institutions to conduct the research.

The mission of RICIS is to conduct, coordinate and disseminate research on computing and information systems among researchers, sponsors and users from UH-Clear Lake, NASA/JSC, and other research organizations. Within UH-Clear Lake, the mission is being implemented through interdisciplinary involvement of faculty and students from each of the four schools: Business, Education, Human Sciences and Humanities, and Natural and Applied Sciences.

Other research organizations are involved via the "gateway" concept. UH-Clear Lake establishes relationships with other universities and research organizations, having common research interests, to provide additional sources of expertise to conduct needed research.

A major role of RICIS is to find the best match of sponsors, researchers and research objectives to advance knowledge in the computing and information sciences. Working jointly with NASA/JSC, RICIS advises on research needs, recommends principals for conducting the research, provides technical and administrative support to coordinate the research, and integrates technical results into the cooperative goals of UH-Clear Lake and NASA/JSC.

ART/Ada Design Project - Phase I
Task 2 Report: Detailed Design

Status Report

Preface

This research was conducted under auspices of the Research Institute for Computing and Information Systems by Inference Corporation. Dr. Charles McKay served as RICIS research coordinator.

Funding has been provided by the Information Systems Directorate, NASA/JSC through Cooperative Agreement NCC 9-16 between the NASA Johnson Space Center and the University of Houston-Clear Lake. The NASA technical monitor for this activity was Robert T. Savely, of the Software Technology Branch, Information Technology Division, Information Systems Directorate, NASA/JSC.

The views and conclusions contained in this report are those of the author and should not be interpreted as representative of the official policies, either express or implied, of NASA or the United States Government.

ART/Ada Design Project - Phase I

Task 2 Report: Detailed Design

Status Report

for

Subcontract 015

RICIS Research Activity SE.19

NASA Cooperative Agreement NCC-9-16

March 1988 - October 1988

Bradley P. Allen

Inference Corporation

5300 W. Century Blvd.

Los Angeles, CA 90045

24 October 88 16:42

Table of Contents

1. Introduction

Under subcontract to University of Houston - Clear Lake as part of the Cooperative Agreement between UHCL and NASA Johnson Space Center, Inference Corporation is conducting an Ada-Based Expert System Building Tool Design Research Project. The goal of the research project is to investigate various issues in the context of the design of an Ada-based expert systems building tool. We are taking the following approach: using an existing successful design as a starting point, analyze the impact of the Ada language and Ada development methodologies on that design, redesign the system in Ada, and analyze its performance using both complexity-theoretic and empirical techniques. The research project will attempt to achieve a comprehensive understanding of the potential for embedding expert systems in Ada systems, for eventual application in future projects.

This research project consists of four discrete tasks and reports analyzing the research results at the end of each task. If the research demonstrates feasibility of the redesign, the design effort will continue into a second phase that will determine the feasibility of the redesign of a larger subset.

In Task 1, we began our effort with an exploration and comparision of design alternatives. Because the research objective is the demonstration of the feasibility of developing expert systems systems in Ada, and because the ART program is an existing example of a development tool for constructing expert systems systems, ART will serve as a baseline for the design.

The Task 1 report identified an architecture for an initial Ada version of ART, *ART/Ada*. and addressed the following topics:

- The methodology followed in the design and implementation of the architecture.
- Differences in the Ada architecture which may limit or expand functionality.
- Language differences which require or allow differentiation between the base software and the Ada version.

In Task 2, we began to further refine the algorithms specified in the overall design, resolving and documenting any open design issues, identifying each system module, documenting the internal architecture and control logic, and describing the primary data structures involved in the module. This data was compiled and provided in this report which constitutes the deliverable for Task 2.

The report, a sequel to the previous report, Prototype Overall Design, describes the detailed design of the ART/Ada prototype that is composed of the Ada Generator and the ART/Ada Runtime System by including the followings:

- The defintion of the user interface for the Ada generator --- The Ada generator is a collection

of C functions that are linked with ART-IM 1.5, and generate the Ada source code that initialize the ART/Ada knowledge base data structure. ART-IM (the Automated Reasoning Tool for Information Management), formerly called ART/C, is an expert system building tool marketed by Inference. Written in C, ART-IM supports the subset of ART features [2] [3].

- The Ada specifications of the Ada source code generated by the Ada generator.
- The examples of the ART/Ada main program.
- The Ada package specifications of the ART/Ada runtime system.
- The Ada package specifications of the Booch Components [1] used to implement basic data structures such as list, string, etc.

In Task 3, we will develop a comprehensive protocol for the analytic and empirical analysis of the performance characteristics of the specified algorithms, and implement in Ada the algorithms necessary to support the analytic and empirical analysis. This protocol will be compiled and provided in a deliverable report.

In Task 4, we will conduct the experiments and analysis defined in the protocol developed in Task 3, and will prepare a final report describing the results. To the extent that the results demonstrate the feasibility of redesign, we will include in this report suggestions for further specification of additional algorithms in a second design phase.

2. Ada Generator

2.1 Introduction

The Ada Generator is an Ada deployment option of ART-IM [3], and is an essential component of the ART/Ada prototype. The generator will be written in C and linked with ART-IM. It should be invoked from the ART-IM command line after an ART program is loaded into ART-IM. Its output is an Ada source code which would be compiled and linked with the ART/Ada runtime system. In essence, the generated Ada code initializes the ART/Ada knowledge base data structures.

In the following sections, the detailed design of the Ada generator will be described as C function definitions. The Ada specifications of the generated Ada code will be also shown.

2.2 Invoking the Ada Generator

The Ada generator traverses through the internal C data structures of ART-IM and generates the Ada code that will initialize the equivalent Ada data structures for ART/Ada at run time.

The Ada generator can be invoked at the ART-IM prompt with the following command:

```
(generate-ada <filename>)
```

where <file-name> is a string representing a file name or a stream. It returns T if succeeds and NIL if fails.

An alternative way is to call a C function, a_generate_ada:

```
boolean a_generate_ada(output)
art_object output;
```

where <file-name> is a string representing a file name or a stream. It returns non-zero if succeeds and zero if fails.

This command will save only the current application, if the system is configured to run multiple applications concurrently. The user must switch to the desired application if another application than the current one should be prepared for deployment.*

*Both ART-IM and ART/Ada are designed to support multiple ART applications and multiple users in a single inference engine executable image.

2.3 Ada Source Code Generated by the Ada Generator

The generated Ada code includes a procedure called INIT which initializes an application in the ART/Ada knowledge base.

Below is the package specification generated by the Ada Generator for an application, APPLICATION_1:

```
-------------------------------------------------------------------
--
--                 C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
-------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: APPLICATION_1
--
-- Function: This package contains subprograms generated by the Ada Generator
--           of ART-IM for an application.
--
-- State Variables:
--           None
-- State Variable Initialization:
--           None
-- Change Log:
--
-------------------------------------------------------------------------
package APPLICATION_1 is

   ----------------------------------------------------------------------
   -- Function: initializes an application in the ART/Ada knowledge base
   ----------------------------------------------------------------------
   procedure INIT;

end APPLICATION_1;
```

In addition to generating a package specification for an application, the Ada generator also generates

the seperate procedure body for INTERPRETER_SUB.FUNCALL. This procedure is the top-level procedure called by the function call interpreter to call out to Ada subprograms. These Ada subprograms consist of those used internally by ART/Ada and those defined by the user. All user-defined Ada subprograms should be defined in the package USER_SUB.

3. ART/Ada Runtime System

3.1 Introduction

The ART/Ada runtime system is composed of Ada packages. These packages, when linked with the Ada code generated by the Ada generator, will yield an executable image tailored for one or more ART applications.

In the following sections, the "with" interdependency of these packages will be depicted in a Booch diagram, and their package specifications will be included.

In addition to these packages, some of the Booch Components [1] are used to implement basic data structures such as lists, strings, stacks, etc. The package specifications of these Booch Components will be also included.

3.2 Booch Diagram of ART/Ada Runtime System

The design of the ART/Ada runtime system follows the object-oriented design(OOD) methodology.

The ART/Ada packages are named based on the following postfix convention:

- ADO - Abstract Data Object
- ADT - Abstract Data Type
- SUB - SUBroutines
- DCL - DeCLarations

The Abstract Data Object is a package that contains encapsulated data and operations (expressed as subprograms) performed upon that data. The data is static and local to that package. The data is known as State Data.

The Abstract Data Type is a package that contains an abstract type and operations performed on that abstract type. The operations are expressed as subprograms and the abstract type is declared as the type of one or more parameters of the subprograms within the package.

The package of subroutines is a package of logically related subroutines. There exists no encapsulated data in this package.

The package of declarations is a package of logically related declarations. These declarations may be types, constants, or exceptions.

The ART/Ada runtime system is composed of public Ada packages and internal Ada packages. The following is the list of public packages that can be with'ed by the user:

- ART: This package is the top-level package that contains public functions that operate on ART/Ada. This package should be always with'ed by the user program.
- ERROR_HDL_SUB: This package contains subprograms for error handling, and the user can redefine the package body to customize the error handlers.
- USER_INTERFACE_SUB: This package contains the command loop. This package is not necessary to with when the presence of the user interface is not needed (i.e. embedded applications).

Other packages are with'ed only by the internal ART/Ada packages and should not be with'ed by the user. The following is the list of the internal packages:

- LEX_ADO: This package contains a lexer for the command loop. It is with'ed only by the USER_INTERFACE_SUB package, and will not be included unless the USER_INTERFACE_SUB package is with'ed by the user.
- PARSER_ADO: This package contains a parser for the command loop. It is with'ed only by the USER_INTERFACE_SUB package, and will not be included unless the USER_INTERFACE_SUB package is with'ed by the user.
- STRUCT_DCL: This package contains declarations of the ART/Ada internal data types.
- ART_OBJECT_SUB: This package contains subprograms that operates on the data type, ART_OBJECT. ART_OBJECT is the basic building block of ART facts.
- DATABASE_SUB: This package contains subprograms for the database management.
- CALLIO_SUB: This package contains subprograms for the Ada call-in/call-out.
- INFER_ENG_SUB: This package contains subprograms to run the inference engine.
- INIT_SUB: This package contains subprograms to initialize ART/Ada and is with'ed only by INFER_ENG_SUB.
- GLOBAL_DCL: This package contains declarations of global data structures for applications and users.
- SYSTEM_DCL: This package contains declarations of system variables.
- IO_SUB: This package contains functions to support input and output.
- PATTERN_NET_SUB: This package contains subprograms for the pattern network interpreter.
- JOIN_NET_SUB: This package contains subprograms for the join network interpreter.
- ART_OBJECT_UTIL_SUB: This package contains utility functions for handling ART_OBJECTs.

- AGENDA_SUB: This package contains subprograms for the agenda.
- INTERPRETER_SUB: This package contains subprograms for the function call interpreter.
- ALLOC_SUB: This package contains subprograms for the memory allocation.
- MACRO_SUB: This package contains subprograms that provides high-level access to fields of the internal data structures. These subprograms are in-lined using the pragma inline for better performance.
- UI_INTERNAL_SUB: This package contains subprograms used by internal packages to print tracing information (e.g. watch facts). If the tracing information is not desired, its body can be stubbed out (i.e. Its subprograms could just contain null, or just return null or equivalent.).
- MATH_SUB: This package contains basic math programs that are used mainly by the rule left-hand side (e.g. +, -, *, /, =, >, <, etc.).
- GC_SUB: This package contains subprograms that are used for garbage collection of ART_OBJECT's.
- COMPILER_SUB: This package contains data types and subprograms that are compiler-specific. The file name for this package varies depending on the compiler. For example, this package for the DEC Ada compiler is in the file, DEC_DCL_.ADA.
- LOGICAL_SUB: This package contains subprograms for the logical dependency. The body of this package will be implemented later.

3.3 Example Main Programs

Two examples of the ART/Ada main programs are included in this section: one that includes the user interface, and one that does not. Although the main program should be defined by the user for each application because the name of the package that contains the application specific procedures varies, it would be easy to modify the standard one supplied by the system.

The following is an example of the main program that includes the user interface by calling the command loop:

```
with ART, USER_INTERFACE_SUB, ERROR_HDL_SUB, APPLICATION_1;
procedure MAIN is
begin
  APPLICATION_1.INIT;
  USER_INTERFACE_SUB.COMMAND_LOOP;
exception
  when CONSTRAINT_ERROR =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.CONSTRAINT_ERR);
  when PROGRAM_ERROR    =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.PROGRAM_ERR);
  when STORAGE_ERROR    =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.STORAGE_ERR);
  when TASKING_ERROR    =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.TASKING_ERR);
end MAIN;
```

This main program loads an expert system application called APPLICATION_1, and prompts the user for a command. The USER_INTERFACE package is with'ed to gain access to the command loop procedure.

The following is an example of the main program that is tailored for an embedded application:

```
with ART, ERROR_HDL_SUB, APPLICATION_1;
procedure MAIN is
begin
  APPLICATION_1.INIT;
  ART.RESET;
  ART.RUN_FOREVER;
exception
  when CONSTRAINT_ERROR =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.CONSTRAINT_ERR);
  when PROGRAM_ERROR    =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.PROGRAM_ERR);
  when STORAGE_ERROR    =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.STORAGE_ERR);
  when TASKING_ERROR    =>
    ERROR_HDL_SUB.PROCESS_ERROR(ERROR_HDL_SUB.TASKING_ERR);
end MAIN;
```

This main program loads, resets, and runs an expert system application called APPLICATION_1. The USER_INTERFACE package is not with'ed by the main program, thereby reducing the size of the run-time executable image.

3.4 Package Specification of the ART/Ada Runtime System

3.4.1 Package Specification of ART

```
------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: ART
--
-- Function: This package contains subprograms for the user to call into
--           ART/Ada.  This package is the top-level public package which
--           contains all the operations on ART/Ada.  This package should
--           be always with'ed in the user's program.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------

with STRUCT_DCL, ART_OBJECT_SUB, DATABASE_SUB, INFER_ENG_SUB, CALLIO_SUB, ALLOC_SUB;
use STRUCT_DCL;
package ART is

   ------------------------------------------------------------
   -- Operations on ART objects
   ------------------------------------------------------------
```

```
-----------------------------------------------------------------------
-- Returns a new, permanent art_object reference to the ART object
-- referred to by reference.  Reference may be either a permanent
-- or automatically allocated art_object.
-----------------------------------------------------------------------
function REGISTER_ART_OBJECT(REFERENCE : ART_OBJECT) return ART_OBJECT
   renames ART_OBJECT_SUB.REGISTER_ART_OBJECT;

-----------------------------------------------------------------------
-- Frees the permanent or temporary reference to an art_object;
-- it is an error to continue to use an art_object after freeing the
-- reference to it.
-----------------------------------------------------------------------
procedure UNREGISTER_ART_OBJECT(REFERENCE : ART_OBJECT)
   renames ART_OBJECT_SUB.UNREGISTER_ART_OBJECT;

-----------------------------------------------------------------------
-- Returns TRUE if the two art_objects X and Y are the same object.
-- EQ and EQUAL are equivalent.
-----------------------------------------------------------------------
function EQ(X: ART_OBJECT;
            Y: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.EQ;

-----------------------------------------------------------------------
-- Returns TRUE if the two art_objects X and Y are the same object.
-- EQ and EQUAL are equivalent.
-----------------------------------------------------------------------
function EQUAL(X: ART_OBJECT;
               Y: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.EQUAL;

-----------------------------------------------------------------------
-- Returns the type of an object, as a symbol.
-----------------------------------------------------------------------
function TYPE_OF(OBJ: ART_OBJECT) return ART_OBJECT
   renames ART_OBJECT_SUB.TYPE_OF;

-----------------------------------------------------------------------
-- A_GENTEMP: Creates a new, previously
-- unused symbol.
-----------------------------------------------------------------------
function GENTEMP(STR : STRING) return ART_OBJECT
   renames ART_OBJECT_SUB.GENTEMP;

-----------------------------------------------------------------------
-- Calls the Ada procedure PROCESS once for each permanent OBJECT
-- that has been allocated passing each permanent art_object
-- as the argument to PROCESS in turn.  If PROCESS returns FALSE
-- at any time, then the iteration is terminated at that point.
-----------------------------------------------------------------------
-- generic
--   with procedure PROCESS (THE_ITEM : in  ART_OBJECT;
--                           CONTINUE : out BOOLEAN);
-- procedure FOR_ALL_PER_ART_OBJECTS;

-----------------------------------------------------------------------
-- Returns an ART_OBJECT for the symbol resulting from performing a
-- intern operation on the string str.  Case is presered in str.
-----------------------------------------------------------------------
function ART_SYMBOL(STR : STRING) return ART_OBJECT
   renames ART_OBJECT_SUB.ART_SYMBOL;

-----------------------------------------------------------
```

```
-- Returns an ART_OBJECT that represents the string specified.
-- Case is preserved.
-------------------------------------------------------------
function ART_STRING(STR : STRING) return ART_OBJECT
   renames ART_OBJECT_SUB.ART_STRING;

-------------------------------------------------------------
-- Returns an ART_OBJECT that represents the number specified.
-------------------------------------------------------------
function ART_INTEGER(NUM: INTEGER) return ART_OBJECT
   renames ART_OBJECT_SUB.ART_INTEGER;

-------------------------------------------------------------
-- Returns an ART_OBJECT that represents the number specified.
-------------------------------------------------------------
function ART_FLOAT(NUM: FLOAT_TYPE) return ART_OBJECT
   renames ART_OBJECT_SUB.ART_FLOAT;

-------------------------------------------------------------
-- Returns a STRING that is the print name of the symbol.
-------------------------------------------------------------
function ADA_SYMBOL(SYMBOL: ART_OBJECT) return STRING
   renames ART_OBJECT_SUB.ADA_SYMBOL;

-------------------------------------------------------------
-- Returns a STRING that represents the ART_OBJECT specified.
-------------------------------------------------------------
function ADA_STRING(STR: ART_OBJECT) return STRING
   renames ART_OBJECT_SUB.ADA_STRING;

-------------------------------------------------------------
-- Returns the number represented by the ART_OBJECT specified.
-------------------------------------------------------------
function ADA_INTEGER(NUM: ART_OBJECT) return INTEGER
   renames ART_OBJECT_SUB.ADA_INTEGER;

-------------------------------------------------------------
-- Returns the number represented by the ART_OBJECT specified.
-------------------------------------------------------------
function ADA_FLOAT(NUM: ART_OBJECT) return FLOAT_TYPE
   renames ART_OBJECT_SUB.ADA_FLOAT;

-------------------------------------------------------------
-- Predicates
-------------------------------------------------------------

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a symbol, otherwise FALSE.
-------------------------------------------------------------------
function SYMBOLP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.SYMBOLP;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a string, otherwise FALSE.
-------------------------------------------------------------------
function STRINGP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.STRINGP;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is an integer, otherwise FALSE.
-------------------------------------------------------------------
function INTEGERP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.INTEGERP;
```

```
-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a float, otherwise FALSE.
-------------------------------------------------------------------
function FLOATP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.FLOATP;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a number, otherwise FALSE.
-------------------------------------------------------------------
function NUMBERP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.NUMBERP;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a sequence, otherwise FALSE.
-------------------------------------------------------------------
function SEQUENCEP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.SEQUENCEP;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a fact, otherwise FALSE.
-------------------------------------------------------------------
function FACTP(OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.FACTP;

-------------------------------------------------------------
-- Fact and Sequence Manipulation
-------------------------------------------------------------

-------------------------------------------------------------------
-- Returns the fact with fact number n.  If no fact has that number,
-- returns NULL.
-------------------------------------------------------------------
function FIND_FACT(N: NATURAL) return ART_OBJECT
   renames DATABASE_SUB.FIND_FACT;

-------------------------------------------------------------
-- Returns the fact number of fact.
-------------------------------------------------------------
function FACT_NUMBER(FACT: ART_OBJECT) return NATURAL
   renames DATABASE_SUB.FACT_NUMBER;

-------------------------------------------------------------------------------------------
-- FOR_ALL_FACTS: Iterates over all the facts in the current database, calling a given
-- procedure once for each fact.
-------------------------------------------------------------------------------------------
-- generic
--   with procedure PROCESS (THE_ITEM : in  ART_OBJECT;
--                           CONTINUE : out BOOLEAN);
-- procedure FOR_ALL_FACTS;

-------------------------------------------------------------
-- Returns length of obj.
-------------------------------------------------------------
function LENGTH(OBJ: ART_OBJECT) return NATURAL
   renames ART_OBJECT_SUB.LENGTH;

-------------------------------------------------------------------
-- Returns the position of the first occurrence of value in obj.  If
-- value is not in obj, returns 0.
-------------------------------------------------------------------
function POSITION(VALUE: ART_OBJECT;
                  OBJ: ART_OBJECT) return NATURAL
   renames ART_OBJECT_SUB.POSITION;
```

```
-------------------------------------------------------------------
-- Returns TRUE if value is in obj, and FALSE otherwise.
-------------------------------------------------------------------
function MEMBER(VALUE: ART_OBJECT;
                OBJ: ART_OBJECT) return BOOLEAN
   renames ART_OBJECT_SUB.MEMBER;

-----------------------------------------------------------
-- Returns the nth element of obj.
-----------------------------------------------------------
function NTH(OBJ: ART_OBJECT;
             INDEX: NATURAL) return ART_OBJECT
   renames ART_OBJECT_SUB.NTH;

-----------------------------------------------------------
-- Fact and Sequence Creation
-----------------------------------------------------------

-------------------------------------------------------------------------
-- Constructs an "empty" fact template and returns a pointer to it as an
-- ART object which may later be asserted.  It is an error to assert a
-- template without inserting something into each of the size slots
-- allocated in it.  All templates are permanent.  Additionally they should
-- not be freed with unregister_art_object.  They should only be freed
-- with free_template
-------------------------------------------------------------------------
function MAKE_TEMPLATE(SIZE: NATURAL) return TEMPLATE_TYPE
   renames CALLIO_SUB.MAKE_TEMPLATE;

-------------------------------------------------------------------------
-- This function sets the element of template specified by
-- index to be value.  It is an error to attempt to modify a fact
-- not created with MAKE_TEMPLATE.  The first element of the fact (the
-- relation) is indexed by index 1.  The other elements of the fact
-- have indices 2 through the length of the fact.
-------------------------------------------------------------------------
procedure SET_NTH(TEMPLATE: in out TEMPLATE_TYPE;
                  INDEX: in INTEGER;
                  VALUE: in ART_OBJECT)
   renames CALLIO_SUB.SET_NTH;

-------------------------------------------------------------------------
-- Frees the TEMPLATE_TYPE template.  It is an error to continue to refer to
-- template after freeing.
-------------------------------------------------------------------------
procedure FREE_TEMPLATE(TEMPLATE: in out TEMPLATE_TYPE)
   renames CALLIO_SUB.FREE_TEMPLATE;

-------------------------------------------------------------------------
-- Asserts a fact from the contents of template into the ART database.
-- Template must be constructed using the functions and macros below
-- prior to assertion.  It is an error to assert a fact with an empty fact
-- slot.  A template may be used for any number of assertions.
-------------------------------------------------------------------------
function ASSERT(TEMPLATE: in TEMPLATE_TYPE) return ART_OBJECT
   renames CALLIO_SUB.ASSERT;

-------------------------------------------------------------------------
-- This function takes a template and returns a sequence matching the
-- template.  The sequence returned will not incorporate or alter the
-- template.
-------------------------------------------------------------------------
function SEQUENCE(TEMPLATE: in TEMPLATE_TYPE) return ART_OBJECT
   renames CALLIO_SUB.SEQUENCE;
```

```
------------------------------------------------------------
-- ART Control
------------------------------------------------------------

------------------------------------------------------------
-- Retracts fact from the ART database.
------------------------------------------------------------
procedure RETRACT(FACT: in out ART_OBJECT)
   renames DATABASE_SUB.RETRACT;

-------------------------------------------------
-- RESET: the purpose of this function is to reset the
--   ART/Ada environment.  The fact table is reset to the deffacts
--   statements, the agenda is cleared, and the join and pattern
--   networks are reset.
-------------------------------------------------
procedure RESET
   renames DATABASE_SUB.RESET;

-----------------------------------------------------------------------
-- Function: Runs the inference engine (match-select-act cycle) LIMIT
--           number of times.  Continue to run until the agenda is
--           empty, until the HALT is encountered on the rhs of a rule,
--           until a salience threshold is reached, or until a breakpoint
--           is triggered.
--
-- Parameters: LIMIT - Number of inference engine cycles. (Or number of rules
--             allowed to fire.
--                    >   0   fire that many rules
--                    =   0   then No rules fire
--                    =  -1   then LIMIT := current default limit
--                    <= -2   fire until agenda becomes empty
-----------------------------------------------------------------------
function RUN(RUN_LIMIT: in INTEGER) return INTEGER
   renames INFER_ENG_SUB.RUN;

-----------------------------------------------------------------------
-- Function: Also, this run procedure runs unless a halt is encountered on
--           the right hand side of a rule or ART.HALT is called from Ada.
--           The procedure does not halt when the agenda is empty but goes
--           into an idle state and remains into that state until the
--           assert or retract operation performed upon the database.
-----------------------------------------------------------------------
procedure RUN_FOREVER
   renames INFER_ENG_SUB.RUN_FOREVER;

-----------------------------------------------------------------------
-- Function: Complete the execution of all rhs actions of the current
--           rule and halts the inference engine.
--
-----------------------------------------------------------------------
procedure HALT
   renames INFER_ENG_SUB.HALT;

-----------------------------------------------------------------------
-- Function: It sets the asynchronous Ada function.
--           The asynchronous function should be defined in the USER_SUB
--           package.  ART/Ada will intern this function and assign it
--           a function ID.
-----------------------------------------------------------------------
procedure SET_ASYNCH_FUN(FUN : STRING)
   renames INFER_ENG_SUB.SET_ASYNCH_FUN;

-----------------------------------------------------------------------
```

```
-- Function: It returns the name of the asynchronous Ada function.
-------------------------------------------------------------------
function GET_ASYNCH_FUN return STRING
   renames INFER_ENG_SUB.GET_ASYNCH_FUN;

-----------------------------------------------------------
-- Returns the minimum salience below which rules may not fire.
-----------------------------------------------------------
function GET_SALIENCE_THRESHOLD return SALIENCE_TYPE
   renames INFER_ENG_SUB.GET_SALIENCE_THRESHOLD;

-------------------------------------------------------------------
-- Set the minimum salience below which rules may not fire.  The constant
-- min_salience may be used to reset salience to the initial default.
-------------------------------------------------------------------
function SET_SALIENCE_THRESHOLD(SALIENCE: in INTEGER) return INTEGER
   renames INFER_ENG_SUB.SET_SALIENCE_THRESHOLD;

-----------------------------------------------------------------------------
-- Returns the default limit on rule firings for run.  If the returned value
-- is negative, the default is to let ART run indefinitely.
-----------------------------------------------------------------------------
function GET_LIMIT_DEFAULT return INTEGER
   renames INFER_ENG_SUB.GET_LIMIT_DEFAULT;

-----------------------------------------------------------
-- Sets the default limit on rule firings for run.  If limit is
-- negative, the default is to let ART run indefinitely.
-----------------------------------------------------------
function SET_LIMIT_DEFAULT(LIMIT: in INTEGER) return INTEGER
   renames INFER_ENG_SUB.SET_LIMIT_DEFAULT;

-------------------------------------------------------------------------
-- Returns a boolean that tells whether ART prints informational messages.
-- TRUE means they are printed; FALSE means they are suppressed.
-------------------------------------------------------------------------
function GET_PRINT_MESSAGES return BOOLEAN
   renames INFER_ENG_SUB.GET_PRINT_MESSAGES;

-----------------------------------------------------------
-- Controls whether ART prints informational messages.  TRUE means
-- to print messages; FALSE to suppress printing of messages.
-- TRUE is the default.
-----------------------------------------------------------
function SET_PRINT_MESSAGES(VALUE: BOOLEAN) return BOOLEAN
   renames INFER_ENG_SUB.SET_PRINT_MESSAGES;

-----------------------------------------------------------
-- Convert INTEGER to BOOLEAN.  If 0, then FALSE. TRUE, otherwise.
-----------------------------------------------------------
function INTEGER_TO_BOOLEAN(STATUS : INTEGER) return BOOLEAN
   renames CALLIO_SUB.INTEGER_TO_BOOLEAN;

-----------------------------------------------------------
-- Convert BOOLEAN to INTEGER.  If TRUE, then 1. If FALSE, then 0.
-----------------------------------------------------------
function BOOLEAN_TO_INTEGER(STATUS : BOOLEAN) return INTEGER
   renames CALLIO_SUB.BOOLEAN_TO_INTEGER;

-------------------------------------------------------------------
-- Function: It frees a sequence.
-------------------------------------------------------------------
procedure FREE_SEQUENCE(X : in out ART_OBJECT)
   renames ALLOC_SUB.FREE_SEQUENCE;
```

```
end ART;
```

3.4.2 Package Specification of ERROR_HDL_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: Jim Badura
--
-- Package: ERROR_HDL_SUB
--
-- Function:  This package contains a procedure that performs error
--            recovery for internal ART errors.
--
-- State Variables: None
--
-- State Variable Initialization: None
--
-- Change Log:
--
------------------------------------------------------------------------
with STRUCT_DCL;
use STRUCT_DCL;

package ERROR_HDL_SUB is

  type ERROR_LOC is (LHS_LOC, RHS_LOC, TOPLEVEL_LOC, ASYNC_LOC);

  type ERROR_TYPE is (CONSTRAINT_ERR,NUMERIC_ERR,PROGRAM_ERR,STORAGE_ERR,
                      TASKING_ERR,
                      INTERNAL_ERR, RETRACT_ERR, INTERPRETER_ERR,
                      USER_DEFINED_ERR);
```

```
   INTERNAL_ERROR : exception;
   RETRACT_ERROR : exception;
   INTERPRETER_ERROR : exception;      -- Error in the interpreter.
   USER_DEFINED_ERROR : exception;     -- The User can use this exception

   ------------------------------------------------------------------------
   -- Function: This procedure invokes the appropriate Ada routine for
   --           recovering from the current internal ART error.
   --           This procedure should be separate.
   --
   -- Parameters: ERROR - The current error being handled
   --
   ------------------------------------------------------------------------
   procedure PROCESS_ERROR(ERROR: in ERROR_TYPE);

   ------------------------------------------------------------------------
   -- Function: This procedure issues a warning message.
   --           This procedure should be separate.
   ------------------------------------------------------------------------
   procedure PROCESS_WARNING;

   ------------------------------------------------------------------------
   -- Function: This procedure stores an error message into a buffer
   --           so that the error message could be printed by PROCESS_ERROR later.
   --
   -- Parameters: MESSAGE - The error message.
   --
   ------------------------------------------------------------------------
   procedure ERROR(MESSAGE: in STRING);

   ------------------------------------------------------------------------
   -- Function: This procedure store an warning message into a buffer
   --           so that the error message could be printed by PROCESS_WARNING.
   --
   -- Parameters: MESSAGE - The warning message.
   --
   ------------------------------------------------------------------------
   procedure WARNING(MESSAGE: in STRING);

end ERROR_HDL_SUB;
```

3.4.3 Package Specification of USER_INTERFACE_SUB

```
-------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
-------------------------------------------------------------------
-- Author: J. T. Badura
--
-- Package: USER_INTERFACE_SUB
--
-- Function: This package contains subprograms that controls the user
--           interface.
--
-- State Variables: None
--
-- State Variable Initialization: None
--
-- Change Log:
--
-------------------------------------------------------------------

with LEX_ADO, PARSER_ADO;

package USER_INTERFACE_SUB is
  ---------------------------------------------------------------------
  -- Function: This procedure invokes the interactive ART/Ada command loop.
  --           This procedure will display the initial banner and repeatly
  --           print the ART/Ada prompt for a ART command.
  --           The command loop should handle watch/unwatch, reset, run,
  --           agenda, facts.
  ---------------------------------------------------------------------
  procedure COMMAND_LOOP;
```

```
end USER_INTERFACE_SUB;
```

3.4.4 Package Specification of LEX_ADO

```
------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author  : James T. Badura
--
-- Package : LEX_ADO
--
-- Function: This package encapsulates objects necessary to perform lexical
--           analysis upon the input stream.  The input stream is divided
--           up into a series of lexeme.  The package also maintains the
--           state of three state variables.
--
-- State Variables:
--           BUFFER   - The command line buffer
--           BPTR     - The buffer index
--           CURR_EOL - The current end of line
--
-- State Variable Initialization:
--           The state variables get initialized when GET_COMMAND_LINE is
--           called.  Initialization is as follows:
--
--           BUFFER(1..80) := (1..80 => ' ')
--           BPTR          := 1
--           CURR_EOL      := LAST; (LAST is returned by TEXT_IO.GET_LINE)
--
-- Change Log:
--           None
------------------------------------------------------------------------------
with TEXT_IO, STRUCT_DCL;
```

```
use  TEXT_IO, STRUCT_DCL;
package  LEX_ADO is
   type LEXEME_TYPE is (LEX_PERIOD,LEX_L_BRACKET,LEX_R_BRACKET,LEX_L_PAREN,
                        LEX_R_PAREN,LEX_EQUALS,LEX_LESS_THAN,LEX_GREATER_THAN,
                        LEX_L_BRACE,LEX_R_BRACE,LEX_ASTERISK,LEX_PLUS,LEX_MINUS,
                        LEX_SLASH,LEX_BACK_SLASH,LEX_TILDE,LEX_COMMA,LEX_COLON,
                        LEX_SEMICOLON,LEX_AT_SIGN,LEX_LOGICAL_OR,LEX_LOGICAL_AND,
                        LEX_NOT_EQUALS,LEX_LESS_THAN_OR_EQUAL,
                        LEX_GREATER_THAN_OR_EQUAL,

                        LEX_BREAK,LEX_UN_BREAK,LEX_WATCH,LEX_UN_WATCH,LEX_RESET,
                        LEX_RUN,LEX_AGENDA,LEX_FACTS,LEX_HALT,LEX_EXIT,LEX_HELP,
                        LEX_SYSTEM,LEX_RULES,LEX_MATCHES,LEX_ACTIVATIONS,
                        LEX_STATUS,LEX_ALL,LEX_PRINT_MEMORY_USAGE,

                        LEX_INTEGER,LEX_REAL,
                        LEX_IDENTIFIER,LEX_CHAR_STRING,LEX_ERROR);

   subtype LEX_OPERATOR is LEXEME_TYPE range LEX_PERIOD..
                                             LEX_GREATER_THAN_OR_EQUAL;

   subtype LEX_RESERVED_WORD is LEXEME_TYPE range LEX_BREAK..LEX_ALL;

   type LEXEME_ITEM_TYPE is
      record
         VAL: LEXEME_TYPE;       -- Lexeme value
         STR: STR_PTR_TYPE;      -- Lexeme string value
      end record;


   ------------------------------------------------------------------------
   --  Gets the next command line into a character buffer of length BUFFER_SIZE.
   --  Sets the state data (refer to state data for more details).
   ------------------------------------------------------------------------
   procedure GET_NEXT_COMMAND_LINE;

   ------------------------------------------------------------------------
   --  Scans for the next lexeme
   ------------------------------------------------------------------------
   procedure SCAN(LEX_ITEM: in out LEXEME_ITEM_TYPE;
                  EOL     : out    BOOLEAN);

end LEX_ADO;
```

3.4.5 Package Specification of PARSER_ADO

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: J. T. Badura
--
-- Package: PARSER_ADO
--
-- Function: This package contains subprograms necessary to perform a
--           top-down recursive descent parsing of input commands of
--           the user interface.
-- State Variables:
--           CURR_LEXEME    - Current lexeme being parsed
--           NEXT_LEXEME    - The next lexeme (one lexeme look-ahead)
--           PARSER_ERRORS - An array of parser errors
-- State Variable Initialization:
--           CURR_LEXEME := first call  of LEX.SCAN
--           NEXT_LEMEME := second call of LEX.SCAN
--           PARSER_ERRORS(1..PARSER_ERRORS_SIZ) := null;
--
-- Change Log:
--
------------------------------------------------------------------------
with LEX_ADO,AGENDA_SUB,DATABASE_SUB,INFER_ENG_SUB;
package PARSER_ADO is
  ----------------------------------------------------------------------
  -- Function: This procedure parses a user interface command using a
  --           recursive descent algorithm.  If an error is encountered,
  --           an error message is written to the screen.
  ----------------------------------------------------------------------
```

```
    procedure PARSE_A_COMMAND;
end PARSER_ADO;
```

3.4.6 Package Specification of STRUCT_DCL

```
-----------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
-----------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: STRUCT_DCL
--
-- Function: This package contains declaration of data structures for
--           the ART/Ada kernel.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
-----------------------------------------------------------------------

--  We are using a variable-length array as a main data structure for
--     rules, LHS patterns, alpha-memory, etc.
--  An example of a variable-length array, String is from Booch:
--
--  type Substring is array(Positive range <>) of Item;
--  type Structure is access Substring;
--  type String is
--      record
--          The_Length : Natural := 0;
```

```
--          The_Items   : Structure;
--      end record;

with TEXT_IO, List_Single_Unbounded_Managed, String_Sequential_Unbounded_Managed_Iterator, COMPI

package STRUCT_DCL is

------------------------------------------------------------------------
--
--      ART DATABASE OBJECTS
--
------------------------------------------------------------------------

   package CHARACTER_STRING is new
   String_Sequential_Unbounded_Managed_Iterator
   (ITEM => STANDARD.CHARACTER,
   SUBSTRING => STANDARD.STRING,
   "<" => "<");

   subtype STR_PTR_TYPE is CHARACTER_STRING.STRING;

   type STR_PTR_ARRAY_TYPE is array(POSITIVE range <>) of STR_PTR_TYPE;

   subtype FLOAT_TYPE is COMPILER_SUB.INTERNAL_FLOAT_TYPE;

   type FLOAT_ARRAY_TYPE is array(POSITIVE range <>) of FLOAT_TYPE;

   type SEQUENCE_TYPE;

   type SEQUENCE_PTR_TYPE is access SEQUENCE_TYPE;

   type FACT_TYPE;

   type FACT_PTR_TYPE is access FACT_TYPE;

   type ELEMENT;

   type ART_OBJECT is access ELEMENT;

   type ART_OBJECT_PTR_TYPE is access ART_OBJECT;

   --
   -- instantiations of generic List package from Booch components
   --
   package ART_OBJECT_LIST is new LIST_SINGLE_UNBOUNDED_MANAGED(ITEM => ART_OBJECT);

   package INTEGER_LIST is new LIST_SINGLE_UNBOUNDED_MANAGED(ITEM => INTEGER);

   subtype ART_OBJECT_LIST_TYPE is ART_OBJECT_LIST.LIST;

   subtype RULE_LIST_TYPE is ART_OBJECT_LIST.LIST;

   subtype INTEGER_LIST_TYPE is INTEGER_LIST.LIST;

   subtype MATCH_LIST_TYPE is INTEGER_LIST.LIST;

   type GLOBAL_VARIABLE is
   record
      NAME: ART_OBJECT;
      RULES: RULE_LIST_TYPE;
      VALUE: ART_OBJECT;
   end record;

   type GLOBAL_VARIABLE_PTR_TYPE is access GLOBAL_VARIABLE;
```

```
type FUNC_TYPE;
type FUNC_PTR_TYPE is access FUNC_TYPE;

type DATA_ENUM_TYPE is (INTEGER_ELEMENT, FLOAT_ELEMENT, FUN_DEF_ELEMENT,
                        STRING_ELEMENT, SYMBOL_ELEMENT, STREAM_ELEMENT,
                        SEQUENCE_ELEMENT, FACT_ELEMENT, VAR_NAME_ELEMENT,
                        GLOBAL_VAR_ELEMENT);

type DATA_SPEC(DATA_CODE: DATA_ENUM_TYPE ) is
record
   case DATA_CODE is
      when INTEGER_ELEMENT => INTEGER_VALUE : INTEGER;
      when FLOAT_ELEMENT => FLOAT_VALUE : FLOAT_TYPE;
      when FUN_DEF_ELEMENT => FUN_DEF: FUNC_PTR_TYPE;
      when STRING_ELEMENT => STRING_VALUE: STR_PTR_TYPE;
      when SYMBOL_ELEMENT => SYMBOL : STR_PTR_TYPE;
      when STREAM_ELEMENT => STREAM : TEXT_IO.FILE_TYPE;  -- Ada generic file_type
      when SEQUENCE_ELEMENT => SEQ : SEQUENCE_PTR_TYPE;
      when FACT_ELEMENT => FACT: FACT_PTR_TYPE;
      when VAR_NAME_ELEMENT => VAR_NAME: ART_OBJECT;
      when GLOBAL_VAR_ELEMENT => GLOBAL: GLOBAL_VARIABLE_PTR_TYPE;
   end case;
end record;

type DATA_SPEC_PTR_TYPE is access DATA_SPEC;

-- The types of art-objects.
type ELEMENT_TYPE is (INTEGER_ELEMENT, FLOAT_ELEMENT, SYMBOL_ELEMENT, STRING_ELEMENT,
  STREAM_ELEMENT, SEQUENCE_ELEMENT, FACT_ELEMENT, FCALL_ELEMENT,
  BVAR_ELEMENT, BVARS_ELEMENT, NOTBVAR_ELEMENT, NOTBVARS_ELEMENT,
  TEMPLATE_ELEMENT, BVAR_DEF_INST_ELEMENT, BVARS_DEF_INST_ELEMENT, NOTBVAR_DEF_INST_ELEMENT,
  NOTBVARS_DEF_INST_ELEMENT, GLOBAL_VAR_ELEMENT, VARIABLE_ELEMENT,
  IO_BUFFER_ELEMENT, INTERNAL_ELEMENT);

type ELEMENT is
record
   KIND: ELEMENT_TYPE;
   PERMANENT_FLAG : BOOLEAN;
   MARKED_FLAG : BOOLEAN;
   HASH_INDEX : NATURAL;
   NEXT: ART_OBJECT;
   DATA: DATA_SPEC_PTR_TYPE;
end record;

type ART_OBJECT_ARRAY_TYPE is array(POSITIVE range <>) of ART_OBJECT;
type ART_OBJECT_ARRAY_PTR_TYPE is access ART_OBJECT_ARRAY_TYPE;

type ART_OBJECT_STACK_TYPE is
   record
      SIZE  : NATURAL := 0;
      STACK : ART_OBJECT_ARRAY_PTR_TYPE;
   end record;

type INTEGER_ARRAY_TYPE is array(POSITIVE range <>) of INTEGER;
type INTEGER_ARRAY_PTR_TYPE is access INTEGER_ARRAY_TYPE;
type INT_STACK_TYPE is
   record
      SIZE  : NATURAL := 0;
      STACK : INTEGER_ARRAY_PTR_TYPE;
   end record;

type SEQUENCE_TYPE is
record
   LENGTH: NATURAL := 0;
```

```
   REF_COUNT: INTEGER;
   DATA: ART_OBJECT_ARRAY_PTR_TYPE;
end record;

subtype TEMPLATE_TYPE is SEQUENCE_PTR_TYPE;

subtype FUNC_ID_TYPE is NATURAL;

type CODE_ELEMENT_TYPE is (BOXED_ELEMENT, OP_CODE_ELEMENT,
                           INDEX_ELEMENT, STRING_ELEMENT);

type VALUE_SPEC_TYPE is (ART_OBJECT_ELEMENT, FLOAT_ELEMENT, BOOLEAN_ELEMENT,
                         INTEGER_ELEMENT, STRING_ELEMENT);

subtype INDEX_TYPE is INTEGER;

type RULEINFO;
type RULEINFO_PTR_TYPE is access RULEINFO;

type VALUE_SPEC is
record
   KIND  : VALUE_SPEC_TYPE;
   INDEX : INDEX_TYPE;
end record;

type VALUE_SPEC_PTR_TYPE is access VALUE_SPEC;
type VALUE_SPEC_ARRAY_TYPE is array(POSITIVE range <>) of VALUE_SPEC;
type VALUE_SPEC_ARRAY_PTR_TYPE is access VALUE_SPEC_ARRAY_TYPE;

type OP_CODE_TYPE is (OP_BRANCH,
--                    OP_BRANCH_IF_TRUE,
--                    OP_BRANCH_IF_FALSE,
                      OP_PUSH_CONSTANT,
                      OP_PUSH_VARIABLE,
                      OP_POP,
                      OP_SAVE_VAR,
                      OP_CALL,
                      OP_ART_OBJ_TO_INT,
--                    OP_ART_OBJ_TO_ART_FLOAT,
                      OP_ART_OBJ_TO_C_FLOAT,
--                    OP_ART_FLOAT_TO_C_FLOAT,
--                    OP_ART_OBJ_TO_ART_SYM,
                      OP_ART_OBJ_SYM_TO_C_STR,
                      OP_ART_SYM_TO_C_STR,
                      OP_ART_OBJ_TO_ART_STR,
                      OP_ART_OBJ_STR_TO_C_STR,
                      OP_ART_STR_TO_C_STR,
                      OP_ART_OBJ_TO_C_OBJ,
                      OP_ART_OBJ_TO_BOOL,
                      OP_ART_OBJ_TO_STREAM,
                      OP_INT_TO_ART_OBJ,
                      OP_BOOL_TO_ART_OBJ,
                      OP_C_OBJ_TO_ART_OBJ,
                      OP_C_FLOAT_TO_ART_OBJ,
                      OP_C_FLOAT_TO_ART_FLOAT,
                      OP_ART_FLOAT_TO_ART_OBJ,
                      OP_C_STR_TO_ART_OBJ_SYM,
                      OP_ART_SYM_TO_ART_OBJ,
                      OP_C_STR_TO_ART_SYM,
                      OP_C_STR_TO_ART_OBJ_STR,
                      OP_ART_STR_TO_ART_OBJ,
                      OP_C_STR_TO_ART_STR,
                      OP_STREAM_TO_ART_OBJ,
                      OP_EXPAND_SEQUENCE,
```

```
                        OP_END,
                        OP_GET_VAR_PAT_STACK,
                        OP_GET_VAR_PAT_VECTOR,
                        OP_GET_VAR_JOIN,
                        OP_GET_FACT_ATOM,
                        OP_INIT_LOCAL,
                        OP_ART_OBJ_TO_FACT,
                        OP_FACT_TO_ART_OBJ,
                        OP_FAST_ASSERT,
                        OP_PUSH_PAT_STACK,
                        OP_PUSH_PAT_VECTOR,
                        OP_PUSH_JOIN,
                        OP_PUSH_FACT_ATOM,
                        OP_CALL_FLOAT,
                        OP_PUSH_DOUBLE,
                        OP_BRANCH_AO_TRUE,
                        OP_BRANCH_BOOL_TRUE,
                        OP_BRANCH_AO_FALSE,
                        OP_BRANCH_BOOL_FALSE,
                        OP_SLOW_ASSERT,
                        OP_ART_OBJ_TO_SEGMENT,
                        OP_SEGMENT_TO_ART_OBJ,
                        OP_INT_TO_ART_FLOAT,
                        OP_INT_TO_C_FLOAT,
                        OP_C_FLOAT_TO_INT,
                        OP_ART_FLOAT_TO_INT,
                        OP_EQ_CONST_VAR_AO,
                        OP_EQ_CONST_STACK_AO,
                        OP_EQ_CONST_STACK_FLOAT,
                        OP_EQ_CONST_STACK_LIT,
                        OP_EQ_VAR_VAR_AO,
                        OP_EQ_VAR_STACK_AO,
                        OP_EQ_STACK_STACK_AO,
                        OP_NOT_BOOL_BOOL,
                        OP_NOT_AO_BOOL,
                        OP_NOT_BOOL_AO,
                        OP_NOT_AO_AO,
                        OP_BOOL_TO_ART_SYM,
                        OP_ART_SYM_TO_BOOL,
                        OP_SAVE_GLOBAL,
                        OP_START_TIME,
                        OP_END_TIME,
                        OP_CALL_DEFUN,
                        OP_DEFUN_END,
                        OP_PUSH_GLOBAL,
                        OP_OPEN_SEGMENT,
                        OP_SAVE_SP,
                        OP_BRANCH_AND_POP_AO_FRAME,
                        OP_SET_AO_FRAME,
                        OP_DEFSCHEMA,
                        OP_SCHEMA_ASSERT,
                        OP_SCHEMA_RETRACT,
                        OP_SCHEMA_MODIFY,
                        OP_BUILD_SUBSEQ);


for OP_CODE_TYPE use (OP_BRANCH => 1,
--                      OP_BRANCH_IF_TRUE => 2,
--                      OP_BRANCH_IF_FALSE => 3,
                        OP_PUSH_CONSTANT => 4,
                        OP_PUSH_VARIABLE => 5,
                        OP_POP => 6,
                        OP_SAVE_VAR => 8,
                        OP_CALL => 10,
```

```
                OP_ART_OBJ_TO_INT => 11,
--              OP_ART_OBJ_TO_ART_FLOAT => 13,
                OP_ART_OBJ_TO_C_FLOAT => 15,
--              OP_ART_FLOAT_TO_C_FLOAT => 17,
--              OP_ART_OBJ_TO_ART_SYM => 18,
                OP_ART_OBJ_SYM_TO_C_STR => 20,
                OP_ART_SYM_TO_C_STR => 22,
                OP_ART_OBJ_TO_ART_STR => 23,
                OP_ART_OBJ_STR_TO_C_STR => 25,
                OP_ART_STR_TO_C_STR => 27,
                OP_ART_OBJ_TO_C_OBJ => 28,
                OP_ART_OBJ_TO_BOOL => 30,
                OP_ART_OBJ_TO_STREAM => 31,
                OP_INT_TO_ART_OBJ => 32,
                OP_BOOL_TO_ART_OBJ => 33,
                OP_C_OBJ_TO_ART_OBJ => 34,
                OP_C_FLOAT_TO_ART_OBJ => 35,
                OP_C_FLOAT_TO_ART_FLOAT =>36,
                OP_ART_FLOAT_TO_ART_OBJ => 37,
                OP_C_STR_TO_ART_OBJ_SYM => 38,
                OP_ART_SYM_TO_ART_OBJ => 39,
                OP_C_STR_TO_ART_SYM => 40,
                OP_C_STR_TO_ART_OBJ_STR => 41,
                OP_ART_STR_TO_ART_OBJ => 42,
                OP_C_STR_TO_ART_STR => 43,
                OP_STREAM_TO_ART_OBJ => 44,
                OP_EXPAND_SEQUENCE => 46,
                OP_END => 47,
                OP_GET_VAR_PAT_STACK => 48,
                OP_GET_VAR_PAT_VECTOR => 49,
                OP_GET_VAR_JOIN => 50,
                OP_GET_FACT_ATOM => 51,
                OP_INIT_LOCAL => 53,
                OP_ART_OBJ_TO_FACT => 55,
                OP_FACT_TO_ART_OBJ => 57,
                OP_FAST_ASSERT => 58,
                OP_PUSH_PAT_STACK => 59,
                OP_PUSH_PAT_VECTOR => 60,
                OP_PUSH_JOIN => 61,
                OP_PUSH_FACT_ATOM => 62,
                OP_CALL_FLOAT => 65,
                OP_PUSH_DOUBLE => 66,
                OP_BRANCH_AO_TRUE => 67,
                OP_BRANCH_BOOL_TRUE => 68,
                OP_BRANCH_AO_FALSE => 69,
                OP_BRANCH_BOOL_FALSE => 70,
                OP_SLOW_ASSERT => 71,
                OP_ART_OBJ_TO_SEGMENT => 72,
                OP_SEGMENT_TO_ART_OBJ => 73,
                OP_INT_TO_ART_FLOAT => 74,
                OP_INT_TO_C_FLOAT => 75,
                OP_C_FLOAT_TO_INT => 76,
                OP_ART_FLOAT_TO_INT => 77,
                OP_EQ_CONST_VAR_AO => 78,
                OP_EQ_CONST_STACK_AO => 79,
                OP_EQ_CONST_STACK_FLOAT => 80,
                OP_EQ_CONST_STACK_LIT => 81,
                OP_EQ_VAR_VAR_AO => 82,
                OP_EQ_VAR_STACK_AO => 83,
                OP_EQ_STACK_STACK_AO => 84,
                OP_NOT_BOOL_BOOL => 85,
                OP_NOT_AO_BOOL => 86,
                OP_NOT_BOOL_AO => 87,
                OP_NOT_AO_AO => 88,
```

```
                        OP_BOOL_TO_ART_SYM => 89,
                        OP_ART_SYM_TO_BOOL => 90,
                        OP_SAVE_GLOBAL => 91,
                        OP_START_TIME => 92,
                        OP_END_TIME => 93,
                        OP_CALL_DEFUN => 94,
                        OP_DEFUN_END => 95,
                        OP_PUSH_GLOBAL => 96,
                        OP_OPEN_SEGMENT => 97,
                        OP_SAVE_SP => 98,
                        OP_BRANCH_AND_POP_AO_FRAME =>99,
                        OP_SET_AO_FRAME => 100,
                        OP_DEFSCHEMA => 101,
                        OP_SCHEMA_ASSERT => 102,
                        OP_SCHEMA_RETRACT => 103,
                        OP_SCHEMA_MODIFY => 104,
                        OP_BUILD_SUBSEQ => 105);

type CODE_ELEMENT(ELEMENT_CODE: CODE_ELEMENT_TYPE) is
record
   case ELEMENT_CODE is
      when BOXED_ELEMENT => BOXED_OBJECT : ART_OBJECT;
      when STRING_ELEMENT => STRING_VALUE : STR_PTR_TYPE;
      when OP_CODE_ELEMENT => OP_CODE: OP_CODE_TYPE;
      when INDEX_ELEMENT => INDEX : INDEX_TYPE;
   end case;
end record;

type CODE_ELEMENT_PTR_TYPE is access CODE_ELEMENT;
type CODE_ELEMENT_PTR_ARRAY_TYPE is array(POSITIVE range <>) of CODE_ELEMENT_PTR_TYPE;
type CODE_ELEMENT_PTR_ARRAY_PTR_TYPE is access CODE_ELEMENT_PTR_ARRAY_TYPE;

-- Contains the relocation information for the code_vector.
-- Index is the location in the code vector that must be relocated.
-- Type is the type of the object to be relocated.
type RELOC_VECTOR is
record
   INDEX : INTEGER;
   KIND : ELEMENT_TYPE;
end record;

type RELOC_VECTOR_PTR_TYPE is access RELOC_VECTOR;
type RELOC_VECTOR_ARRAY_TYPE is array(POSITIVE range <>) of RELOC_VECTOR;
type RELOC_VECTOR_ARRAY_PTR_TYPE is access RELOC_VECTOR_ARRAY_TYPE;

-- The outer structure for a code vector.  Points to the code_vector and
-- to the vector of relocation information.

type ART_CODE is
record
   CODE_VECTOR : CODE_ELEMENT_PTR_ARRAY_PTR_TYPE;
   RELOC_VECTOR_VALUE : RELOC_VECTOR_ARRAY_PTR_TYPE;
end record;

type ART_CODE_PTR_TYPE is access ART_CODE;
-- type ART_CODE_ARRAY_TYPE is array(POSITIVE range <>) of ART_CODE;
-- type ART_CODE_ARRAY_PTR_TYPE is access ART_CODE_ARRAY_TYPE;

type LOGIC is
record
   LEFT_FACT: INTEGER;
   LEFT_ELEMENT: INTEGER;
   RIGHT_ELEMENT: INTEGER;
end record;
```

```
type LOGIC_PTR_TYPE is access LOGIC;
type LOGIC_ARRAY_TYPE is array(POSITIVE range <>) of LOGIC;
type LOGIC_ARRAY_PTR_TYPE is access LOGIC_ARRAY_TYPE;

type VAR_MATCH is
record
   LENGTH : INTEGER;
   MATCHES : LOGIC_ARRAY_PTR_TYPE;
end record;

type VAR_MATCH_PTR_TYPE is access VAR_MATCH;
type VAR_MATCH_ARRAY_TYPE is array(POSITIVE range <>) of VAR_MATCH;
type VAR_MATCH_ARRAY_PTR_TYPE is access VAR_MATCH_ARRAY_TYPE;

type RELATION;
type RELATION_PTR_TYPE is access RELATION;

type TEST_FLAG_TYPE is (TEST_CONSTANT, TEST_EXPRESSION);

type PARENT_NODE_TYPE is (RELATION_PARENT, PATTERN_NODE_PARENT);

type PATTERN_NODE(TEST_FLAG : TEST_FLAG_TYPE;
                  PARENT_FLAG : PARENT_NODE_TYPE;
                  STOP_FLAG : BOOLEAN);

type PATTERN_NODE_PTR_TYPE is access PATTERN_NODE;

type RELATION is
record
   NAME: ART_OBJECT;
   RESTRICTIONS: PATTERN_NODE_PTR_TYPE;
   NEXT: RELATION_PTR_TYPE;
end record;

type RELATION_PTR_ARRAY_TYPE is array(POSITIVE range <>) of RELATION_PTR_TYPE;

type RULEINFO_OR_CODE_ELEMENT_TYPE is (RULEINFO_ELEMENT, ART_CODE_ELEMENT);
type RULEINFO_OR_CODE_TYPE (FLAG : RULEINFO_OR_CODE_ELEMENT_TYPE) is
  record
    case FLAG is
      when RULEINFO_ELEMENT => RULEINFO_VALUE : RULEINFO_PTR_TYPE;
      when ART_CODE_ELEMENT => CODE_VALUE     : ART_CODE_PTR_TYPE;
    end case;
  end record;
type RULEINFO_OR_CODE_PTR_TYPE is access RULEINFO_OR_CODE_TYPE;
type JOIN_NODE;
type JOIN_NODE_PTR_TYPE is access JOIN_NODE;

-------------------------------------------------------------------------
-- JOIN_NODE STRUCTURE:
-- up is a backward chain of join nodes.
-- down chains together the join nodes that depend on this node; when
-- it's zero, eval actually contains a ruleinfo structure for the rule
-- to fire.
-------------------------------------------------------------------------

type JOIN_NODE is
record
   INITIAL_JOIN_FLAG : BOOLEAN;
   POSITIVE_JOIN_FLAG : BOOLEAN;
   -- If this is a negative fact pattern, and the first
   -- fact pattern in the rule, set the flag.
   LEADING_NOT_FLAG : BOOLEAN;
   DEPTH : INTEGER;                          -- The number of patterns processed before join
```

```
   HASH_INDEX : INTEGER;

   EVAL: RULEINFO_OR_CODE_PTR_TYPE;
   -- An expression gathered from succeeding TEST constructs in the case of a NOT node
   -- If the node is not a NOT node, then it is NULL.
   NOT_TESTS: ART_CODE_PTR_TYPE;

   MATCH_LOGIC: VAR_MATCH_PTR_TYPE;   -- variable length array

   RIGHT: JOIN_NODE_PTR_TYPE;              -- The joins on same level
   UP: JOIN_NODE_PTR_TYPE;                 -- parent join
   DOWN: JOIN_NODE_PTR_TYPE;               -- joins which depend on this node
   NEXT_STOP: JOIN_NODE_PTR_TYPE;          -- list of join nodes feeding off a stop node
   ENTRY_PAT: PATTERN_NODE_PTR_TYPE;
end record;

type RHS_BIND;
type RHS_BIND_PTR_TYPE is access RHS_BIND;
type RHS_BIND_PTR_ARRAY_TYPE;
type RHS_BIND_PTR_ARRAY_PTR_TYPE is access RHS_BIND_PTR_ARRAY_TYPE;

type DEPENDENT;
type DEPENDENT_PTR_TYPE is access DEPENDENT;
type DEPENDENT is
record
   NEXT : DEPENDENT_PTR_TYPE;
   DATA : ART_OBJECT;
end record;

type ACTIVATION;
type ACTIVATION_PTR_TYPE is access ACTIVATION;

type PART_MATCH (LAST_JOIN : BOOLEAN);
type PART_MATCH_PTR_TYPE is access PART_MATCH;
type PART_MATCH_PTR_ARRAY_TYPE is array(POSITIVE range <>) of PART_MATCH_PTR_TYPE;
type PART_MATCH_PTR_ARRAY_PTR_TYPE is access PART_MATCH_PTR_ARRAY_TYPE;
type PART_MATCH (LAST_JOIN : BOOLEAN) is
record
   COUNT: INTEGER;
   DEPENDENTS: DEPENDENT_PTR_TYPE;
   JOIN: JOIN_NODE_PTR_TYPE;
   NEXT: PART_MATCH_PTR_TYPE;
   BIND_LENGTH: NATURAL;
   BINDS: RHS_BIND_PTR_ARRAY_PTR_TYPE;
   case LAST_JOIN is
     when TRUE  => ACT : ACTIVATION_PTR_TYPE := null;
     when FALSE => null;
   end case;
end record;

type LOG_SUPPORT;
type LOG_SUPPORT_PTR_TYPE is access LOG_SUPPORT;
-- LOGICAL SUPPORT STRUCTURE
-- List of joins supporting a fact
type LOG_SUPPORT is
record
SUPPORT: PART_MATCH_PTR_TYPE;
NEXT: LOG_SUPPORT_PTR_TYPE;
end record;

-------------------------------------------------------------------------
-- PATTERN_NODE: This data structure describes the actions and characteristics of
-- the pattern network. It contains instructions for exactly what actions should
-- be performed at that node and what kind of node it is.
```

```
--------------------------------------------------------------------------

type TEST_NODE_TYPE (TEST_FLAG : TEST_FLAG_TYPE) is
record
   case TEST_FLAG is
      -- A constant to be tested against.
      when TEST_CONSTANT => CONSTANT_VALUE : ART_OBJECT;

      -- A code vector to be interpreted for the result.
      when TEST_EXPRESSION => EXPRESSION : ART_CODE_PTR_TYPE;

   end case;
end record;

type PARENT_TYPE (PARENT_FLAG : PARENT_NODE_TYPE) is
record
   case PARENT_FLAG is
      -- The relation for a fact START node.
      when RELATION_PARENT => RELATION : RELATION_PTR_TYPE;
      -- The immediate mother pattern node.
      when PATTERN_NODE_PARENT => UP : PATTERN_NODE_PTR_TYPE;
   end case;
end record;

type PATTERN_NODE(TEST_FLAG : TEST_FLAG_TYPE;
   PARENT_FLAG : PARENT_NODE_TYPE;
   STOP_FLAG : BOOLEAN) is
record
   TYPE_SINGLE_FLAG : BOOLEAN; -- These four mutually exclusive
   TYPE_MULTIPLE_FLAG : BOOLEAN;

   TYPE_EMPTY_FLAG : BOOLEAN; -- if this node is only an into or out node

   ACTION_INTO_FLAG : BOOLEAN;
   ACTION_OUT_FLAG : BOOLEAN;
   ACTION_SAME_FLAG : BOOLEAN;
   ACTION_NEXT_SLOT_FLAG : BOOLEAN;
   ACTION_NEXT_VARIABLE_SLOT_FLAG : BOOLEAN;

   TEST_NOTHING_FLAG : BOOLEAN;

   BIND_FLAG : BOOLEAN;

   LEFT : PATTERN_NODE_PTR_TYPE;        -- The sister to the left.
   RIGHT : PATTERN_NODE_PTR_TYPE;       -- The sister to the right.

   TEST_NODE : TEST_NODE_TYPE( TEST_FLAG );

   PARENT: PARENT_TYPE( PARENT_FLAG );

   case STOP_FLAG is
      -- The start of the daughter chain.
      when FALSE => DOWN : PATTERN_NODE_PTR_TYPE;
      -- A list of those join nodes feeding off of this STOP node.
      when TRUE => PATH : JOIN_NODE_PTR_TYPE;
   end case;
end record;

type PAT_REF;
type PAT_REF_PTR_TYPE is access PAT_REF;
type PAT_REF is
record
   PAT : PATTERN_NODE_PTR_TYPE;
   NEXT : PAT_REF_PTR_TYPE;
```

```
end record;

type JOIN_REF;
type JOIN_REF_PTR_TYPE is access JOIN_REF;
type JOIN_REF is
record
   PATH : JOIN_NODE_PTR_TYPE;
   NEXT : JOIN_REF_PTR_TYPE;
end record;

-------------------------------------------------------------------------
-- RHS_BIND: What ART/LISP calls an "instantiation", this
-- describes the items that were used in a particular
-- output of the fact pattern net.
-------------------------------------------------------------------------

type RHS_BIND is
record
   LAST_PAT_NODE : PATTERN_NODE_PTR_TYPE;
   NEXT : RHS_BIND_PTR_TYPE;
   ITEM_LENGTH : INTEGER;
   ITEMS : ART_OBJECT_ARRAY_PTR_TYPE;
end record;

type RHS_BIND_PTR_ARRAY_TYPE is array(POSITIVE range <>) of RHS_BIND_PTR_TYPE;

type FACT_TYPE is
record
   ID: NATURAL;
   BIND_LIST: RHS_BIND_PTR_TYPE;
   SUPPORT: LOG_SUPPORT_PTR_TYPE;
   PROPOSITION: ART_OBJECT;
end record;

-------------------------------------------------------------------------
--
--   FUNCTIONS
--
-------------------------------------------------------------------------

type TEST_TYPE;
type TEST_PTR_TYPE is access TEST_TYPE;
type TEST_TYPE is
record
   VALUE: ART_OBJECT;
   ARG_LIST: TEST_PTR_TYPE;
   NEXT_ARG: TEST_PTR_TYPE;
end record;

type PARAMETER;
type PARAMETER_PTR_TYPE is access PARAMETER;
type PARAMETER is
record
   ARG_TYPE: ART_OBJECT;
   CLASS: ART_OBJECT;
   NEXT: PARAMETER_PTR_TYPE;
   OPTIONAL: TEST_PTR_TYPE;
end record;

type FUNC_TYPE is
record
   ID: FUNC_ID_TYPE;              -- Function ID
   ART_NAME: ART_OBJECT;          -- ART function name
   ADA_NAME: ART_OBJECT;          -- Entry point name
```

```
   RET_TYPE: ART_OBJECT;          -- Return data type
   DEFUN: ART_CODE_PTR_TYPE,
   PARAMS: PARAMETER_PTR_TYPE;    -- Function parameters
   MIN_ARGS: ART_OBJECT;          -- Minimum number of parameters
   MAX_ARGS: ART_OBJECT;          -- Maximum number of parameters
end record;

-------------------------------------------------------------------------
--
--   RULES
--
-------------------------------------------------------------------------

subtype SALIENCE_TYPE is integer range -10000..10000;

type RULEINFO is
record
   NAME: ART_OBJECT;
   DOC_STRING: ART_OBJECT;
   BREAKPOINT_SET : BOOLEAN;

   -- if 1, this is the original rule;
   -- if not, one of the OR derivatives
   COPY_NUMBER: POSITIVE;

   SALIENCE: SALIENCE_TYPE;
   SALIENCE_CODE: ART_CODE_PTR_TYPE;
   RHS_CODE: ART_CODE_PTR_TYPE;

   JOIN_NODE: JOIN_NODE_PTR_TYPE;
   LOGICAL_JOIN: JOIN_NODE_PTR_TYPE;
   NEXT: RULEINFO_PTR_TYPE;
end record;

-------------------------------------------------------------------------
--
--   ACTIVATION
--
-------------------------------------------------------------------------

type ACTIVATION is
record
   RULE: RULEINFO_PTR_TYPE;
   BINDINGS: PART_MATCH_PTR_TYPE;
   SALIENCE: SALIENCE_TYPE;
   NEXT: ACTIVATION_PTR_TYPE;
   PREV: ACTIVATION_PTR_TYPE;
end record;

-- DEFFACT STRUCTURE:  Stores information about a deffacts construct.
--   Name:        The name of the deffacts construct.
--   Description: The string describing the deffacts construct. Used
--                 after the deffacts name.
--   Flist:       Pointer to the list of facts defined by this
--                 deffacts construct.
--   Next:        Pointer to the next deffacts definition structure.

type DEFFACT;
type DEFFACT_PTR_TYPE is access DEFFACT;
type DEFFACT is
record
   NAME: ART_OBJECT;
   DESCRIPTION: ART_OBJECT;
   FLIST: ART_OBJECT_LIST_TYPE;
```

```
        NEXT: DEFFACT_PTR_TYPE;
      end record;

  end STRUCT_DCL;
```

3.4.7 Package Specification of ART_OBJECT_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: ART_OBJECT_SUB
--
-- Function: This package contains subprograms necessary to manage
--           the ART/Ada objects.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
--   Author     Date        Change
--
--   jtb      09/14/88      Added function GET_SYMBOL_GLOBAL_VAL
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package ART_OBJECT_SUB is

   ---------------------------------------------------------------------
```

```
-- Operations on ART objects
-------------------------------------------------------------------------------

-------------------------------------------------------------------
-- Returns a new, permanent art_object reference to the ART object
-- referred to by reference.  Reference may be either a permanent
-- or automatically allocated art_object.
-------------------------------------------------------------------
function REGISTER_ART_OBJECT(REFERENCE : ART_OBJECT) return ART_OBJECT;

-------------------------------------------------------------------
-- Frees the permanent or temporary reference to an art_object;
-- it is an error to continue to use an art_object after freeing the
-- reference to it.
-------------------------------------------------------------------
procedure UNREGISTER_ART_OBJECT(REFERENCE : ART_OBJECT);

-------------------------------------------------------------------
-- Returns TRUE if the two art_objects X and Y are the same object.
-- EQ and EQUAL are equivalent.
-------------------------------------------------------------------
function EQ(X: ART_OBJECT;
            Y: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------------
-- Returns TRUE if the two art_objects X and Y are the same object.
-- EQ and EQUAL are equivalent.
-------------------------------------------------------------------
function EQUAL(X: ART_OBJECT;
               Y: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------------
-- Returns the type of an object, as a symbol.
-------------------------------------------------------------------
function TYPE_OF(OBJ: ART_OBJECT) return ART_OBJECT;

-------------------------------------------------------------------
-- Returns an ART_OBJECT for the symbol resulting from performing a
-- intern operation on the string str.  Case is presered in str.
-------------------------------------------------------------------
function ART_SYMBOL(A_STR : STRING) return ART_OBJECT;

-------------------------------------------------------------
-- Returns an ART_OBJECT that represents the string specified.
-- Case is preserved.
-------------------------------------------------------------
function ART_STRING(A_STR : STRING) return ART_OBJECT;

-------------------------------------------------------------
-- Returns an ART_OBJECT that represents the number specified.
-------------------------------------------------------------
function ART_INTEGER(NUM: INTEGER) return ART_OBJECT;

-------------------------------------------------------------
-- Returns an ART_OBJECT that represents the number specified.
-------------------------------------------------------------
function ART_FLOAT(NUM: FLOAT_TYPE) return ART_OBJECT;

-------------------------------------------------------------
-- Returns a STRING that is the print name of the symbol.
-------------------------------------------------------------
function ADA_SYMBOL(SYMBOL: ART_OBJECT) return STRING;

-------------------------------------------------------------
```

```
-- Returns a STRING that represents the ART_OBJECT specified.
-------------------------------------------------------------
function ADA_STRING(STR: ART_OBJECT) return STRING;

-------------------------------------------------------------
-- Returns the number represented by the ART_OBJECT specified.
-------------------------------------------------------------
function ADA_INTEGER(NUM: ART_OBJECT) return INTEGER;

-------------------------------------------------------------
-- Returns the number represented by the ART_OBJECT specified.
-------------------------------------------------------------
function ADA_FLOAT(NUM: ART_OBJECT) return FLOAT_TYPE;

-------------------------------------------------------------
-- Returns length of obj.
-------------------------------------------------------------
function LENGTH(OBJ: ART_OBJECT) return NATURAL;

-------------------------------------------------------------------
-- Returns the position of the first occurrence of value in obj.  If
-- value is not in obj, returns 0.
-------------------------------------------------------------------
function POSITION(VALUE: ART_OBJECT;
                  OBJ: ART_OBJECT) return NATURAL;

-------------------------------------------------------------------
-- Returns TRUE if value is in obj, and FALSE otherwise.
-------------------------------------------------------------------
function MEMBER(VALUE: ART_OBJECT;
                OBJ: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------
-- Returns the nth element of obj.
-------------------------------------------------------------
function NTH(OBJ: ART_OBJECT;
             INDEX: NATURAL) return ART_OBJECT;

-----------------------------------------------------------------------------
-- Predicates
-----------------------------------------------------------------------------

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a symbol, otherwise FALSE.
-------------------------------------------------------------------
function SYMBOLP(OBJ: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a string, otherwise FALSE.
-------------------------------------------------------------------
function STRINGP(OBJ: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is an integer, otherwise FALSE.
-------------------------------------------------------------------
function INTEGERP(OBJ: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a float, otherwise FALSE.
-------------------------------------------------------------------
function FLOATP(OBJ: ART_OBJECT) return BOOLEAN;

-------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a number, otherwise FALSE.
```

```
-----------------------------------------------------------------------
function NUMBERP(OBJ: ART_OBJECT) return BOOLEAN;

-----------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a sequence, otherwise FALSE.
-----------------------------------------------------------------------
function SEQUENCEP(OBJ: ART_OBJECT) return BOOLEAN;

-----------------------------------------------------------------------
-- Returns TRUE if the ART_OBJECT, OBJ, is a fact, otherwise FALSE.
-----------------------------------------------------------------------
function FACTP(OBJ: ART_OBJECT) return BOOLEAN;

-----------------------------------------------------------------------------
-- HASHING SUBPROGRAMS
-----------------------------------------------------------------------------

-----------------------------------------------------------------------
-- FIND_SYMBOL:  Searches for the string in the hash table. If the
--   string is already in the hash table, then the address of the
--   string is returned.  Otherwise, NULL is returned.
-----------------------------------------------------------------------
function FIND_SYMBOL(STR: STR_PTR_TYPE) return ART_OBJECT;

-----------------------------------------------------------------------
-- init_hash: Purpose is to initialize the hash tables to NULL
-----------------------------------------------------------------------
procedure INIT_HASH;

-----------------------------------------------------------------------
-- HASH:  Computes the hash table location for the given string.
-----------------------------------------------------------------------
function HASH(VALUE: NATURAL;
              TABLE_LENGTH : INTEGER) return INTEGER;

-----------------------------------------------------------------------
-- AO_HASH:  Hashes an ART_OBJECT.
-----------------------------------------------------------------------
procedure AO_HASH( CODE : INTEGER;
                   ADDR : ART_OBJECT);

function PART_MATCH_RIGHT_HASH (RHS_BIND  : in RHS_BIND_PTR_TYPE;
                                JOIN      : in JOIN_NODE_PTR_TYPE;
                                MAT       : in VAR_MATCH_PTR_TYPE) return NATURAL;

function PART_MATCH_LEFT_HASH (RHS_BIND  : in RHS_BIND_PTR_ARRAY_PTR_TYPE;
                               JOIN      : in JOIN_NODE_PTR_TYPE;
                               MAT       : in VAR_MATCH_PTR_TYPE) return NATURAL;

-----------------------------------------------------------------------------
-- miscellaneous subprograms
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
-- A_FREE_SEQUENCE: Decrements a sequence's reference count and deallocates the
-- sequence if the reference count goes to 0 (unused sequence.)
-----------------------------------------------------------------------------
procedure FREE_SEQUENCE(SEQ: ART_OBJECT);

-----------------------------------------------------------------------
-- A_GENTEMP: Creates a new, previously
-- unused symbol.
-----------------------------------------------------------------------
function GENTEMP(STR : STRING) return ART_OBJECT;
```

```
    ------------------------------------------------------------------------------
    -- SEQ_LIST_INTERN: Takes a list of art-object references and nondestructively copies it
    -- into an interned sequence.
    ------------------------------------------------------------------------------
    function SEQ_LIST_INTERN(SEQ_ITEM_LIST: ART_OBJECT_LIST_TYPE) return ART_OBJECT;

    ------------------------------------------------------------------------------
    -- SEQ_INTERN: Interns a sequence of art-objects into the sequence hash table. Maintains
    -- the reference counts of the sequences for garbage collection and freeing.
    ------------------------------------------------------------------------------
    function SEQ_INTERN(VECTOR: ART_OBJECT_ARRAY_PTR_TYPE;
                        OFFSET: INTEGER;
                        LENGTH: INTEGER) return ART_OBJECT;

    ------------------------------------------------------------------------------
    -- SEQ_INTERN: Interns a sequence of art-objects into the sequence hash table. Maintains
    -- the reference counts of the sequences for garbage collection and freeing.
    ------------------------------------------------------------------------------
    function SEQ_INTERN(VECTOR: CODE_ELEMENT_PTR_ARRAY_PTR_TYPE;
                        OFFSET: INTEGER;
                        LENGTH: INTEGER) return ART_OBJECT;

    ------------------------------------------------------------------------------
    -- SYMBOL_INTERN: Interns a symbol into the sequence hash table. Maintains
    -- the reference counts of the sequences for garbage collection and freeing.
    ------------------------------------------------------------------------------
    function SYMBOL_INTERN(SYMBOL: ART_OBJECT) return ART_OBJECT;

    ------------------------------------------------------------------
    -- GET_SYMBOL_GLOVAL_VAL: Returns the global variable associated with the
    --                        symbol given, returns null if no global variable
    --                        exists with that name.
    ------------------------------------------------------------------
    function GET_SYMBOL_GLOBAL_VAL(NAME: ART_OBJECT) return ART_OBJECT;

end ART_OBJECT_SUB;
```

3.4.8 Package Specification of DATABASE_SUB

```
-------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
-------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: DATABASE_SUB
--
-- Function: This package contains subprograms necessary to manage
--           the ART/Ada database.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--      9-14-88 SDL - add offset to seq_intern as the second argument.
--
-------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package DATABASE_SUB is

   ------------------------------------------------
   -- RESET: the purpose of this function is to reset the
   --   ART/Ada environment.  The fact table is reset to the deffacts
   --   statements, the agenda is cleared, and the join and pattern
```

```
    --   networks are reset.
    ------------------------------------------------
    procedure RESET;

    ------------------------------------------------------------------
    -- RMV_OLD_FACTS:  Returns facts that have been retracted to the
    --   pool of available memory.  It is necessary to postpone
    --   returning the facts to memory because rhs actions retrieve
    --   their variable bindings directly from the fact data structure.
    ------------------------------------------------------------------
    procedure RMV_OLD_FACTS;

    ------------------------------------------------------------------------
    -- RETRACT: A public function to retract a fact.
    ------------------------------------------------------------------------
    procedure RETRACT(FACT: in out ART_OBJECT);

    ------------------------------------------------------------------
    -- FIND_FACT:
    -- The function find_fact finds a fact with a given id number.
    ------------------------------------------------------------------
    function FIND_FACT(ID: INTEGER) return ART_OBJECT;

    ------------------------------------------------------------------
    -- FACT_NUMBER:
    -- The function fact_number returns a fact id number given a fact.
    ------------------------------------------------------------------
    function FACT_NUMBER(FACT: ART_OBJECT) return NATURAL;

    ------------------------------------------------
    -- FACTS: Prints out a list of the current facts
    -- to the screen, in sorted order and with a
    -- sum of the total current facts.
    ------------------------------------------------
    procedure FACTS;

    ------------------------------------------------------------------------------
    -- ENTER_FACT: Puts a fact into the data base and sends it through the networks
    ------------------------------------------------------------------------------
    function ENTER_FACT(PROPOSITION: ART_OBJECT) return ART_OBJECT;


    ------------------------------------------------------------------------
    --
    -- REMOVE_RHS_BIND: Removes a RHS bind and updates the sequence reference
    --                  counts of its items.
    --
    ------------------------------------------------------------------------
    procedure REMOVE_RHS_BIND(THIS_BIND: in out RHS_BIND_PTR_TYPE);

end DATABASE_SUB;
```

3.4.9 Package Specification of CALLIO_SUB

```
-- Author: S. Daniel Lee
--
-- Package: CALLIO_SUB
--
-- Function: This package contains subprograms necessary to support
--           call-in.

--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
-------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package CALLIO_SUB is

   ----------------------------------------------------------------------
   -- Constructs an "empty" fact template and returns a pointer to it as an
   -- ART object which may later be asserted.  It is an error to assert a
   -- template without inserting something into each of the size slots
```

```
   -- allocated in it.  All templates are permanent.  Additionally they should
   -- not be freed with unregister_art_object.  They should only be freed
   -- with free_template
   ------------------------------------------------------------------------
   function MAKE_TEMPLATE(SIZE: NATURAL) return TEMPLATE_TYPE;

   ------------------------------------------------------------------------
   -- This function sets the element of template specified by
   -- index to be value.  It is an error to attempt to modify a fact
   -- not created with MAKE_TEMPLATE.  The first element of the fact (the
   -- relation) is indexed by index 1.  The other elements of the fact
   -- have indices 2 through the length of the fact.
   ------------------------------------------------------------------------
   procedure SET_NTH(TEMPLATE: in out TEMPLATE_TYPE;
                     INDEX: in INTEGER;
                     VALUE: in ART_OBJECT);

   ------------------------------------------------------------------------
   -- Frees the TEMPLATE_TYPE template.  It is an error to continue to refer to
   -- template after freeing.
   ------------------------------------------------------------------------
   procedure FREE_TEMPLATE(TEMPLATE: in out TEMPLATE_TYPE);

   ------------------------------------------------------------------------
   -- Asserts a fact from the contents of template into the ART database.
   -- Template must be constructed using the functions and macros below
   -- prior to assertion.  It is an error to assert a fact with an empty fact
   -- slot.  A template may be used for any number of assertions.
   ------------------------------------------------------------------------
   function ASSERT(TEMPLATE: in TEMPLATE_TYPE) return ART_OBJECT;

   ------------------------------------------------------------------------
   -- This function takes a template and returns a sequence matching the
   -- template.  The sequence returned will not incorporate or alter the
   -- template.
   ------------------------------------------------------------------------
   function SEQUENCE(TEMPLATE: in TEMPLATE_TYPE) return ART_OBJECT;

   ------------------------------------------------------------------
   -- Convert INTEGER to BOOLEAN.  If 0, then FALSE. TRUE, otherwise.
   ------------------------------------------------------------------
   function INTEGER_TO_BOOLEAN(STATUS : INTEGER) return BOOLEAN;

   ------------------------------------------------------------------
   -- Convert BOOLEAN to INTEGER.  If TRUE, then 1. If FALSE, then 0.
   ------------------------------------------------------------------
   function BOOLEAN_TO_INTEGER(STATUS : BOOLEAN) return INTEGER;

end CALLIO_SUB;
```

3.4.10 Package Specification of INFER_ENG_SUB

```
------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: J. T. Badura
--
-- Package: INFER_ENG_SUB
--
-- Function: This package contains subprograms that control functions
--           of the inference engine.
--
-- State Variables:
--           None
-- State Variable Initialization:
--           None
-- Change Log:
--
------------------------------------------------------------------------
with STRUCT_DCL;
use  STRUCT_DCL;
package INFER_ENG_SUB is
   ---------------------------------------------------------------------
   -- Function: This procedure initializes ART.  This procedure is called
   --           before any other ART procedures.
   ---------------------------------------------------------------------
   procedure INIT;

   ---------------------------------------------------------------------
   -- Function: Runs the inference engine (match-select-act cycle) LIMIT
   --           number of times.  Continue to run until the agenda is
```

```
--           empty, until the HALT is encountered on the rhs of a rule,
--           until a salience threshold is reached, or until a breakpoint
--           is triggered.
--
-- Parameters: LIMIT - Number of inference engine cycles. (Or number of rules
--              Returns  the number of rules fired.
--              allowed to fire.
--                     >   0   fire that many rules
--                     =   0   then No rules fire
--                     =  -1   then LIMIT := current default limit
--                     <= -2   fire until agenda becomes empty
-------------------------------------------------------------------------
function RUN(RUN_LIMIT: in INTEGER) return INTEGER;

-------------------------------------------------------------------------
-- Function: Also, this run procedure runs unless a halt is encountered on
--           the right hand side of a rule or ART.HALT is called from Ada.
--           The procedure does not halt when the agenda is empty but goes
--           into an idle state and remains into that state until the
--           assert or retract operation performed upon the database.
-------------------------------------------------------------------------
procedure RUN_FOREVER;

-------------------------------------------------------------------------
-- Function: It sets the asynchronous Ada function.
--           The asynchronous function should be defined in the USER_SUB
--           package.  ART/Ada will intern this function and assign it
--           a function ID.
-------------------------------------------------------------------------
procedure SET_ASYNCH_FUN(FUN : STRING);

-------------------------------------------------------------------------
-- Function: It returns the name of the asynchronous Ada function.
-------------------------------------------------------------------------
function GET_ASYNCH_FUN return STRING;

-------------------------------------------------------------------------
-- Function: Returns the current salience threshold; any rules with
--           salience below this threshold are prevented from firing.
--
-------------------------------------------------------------------------
function GET_SALIENCE_THRESHOLD return SALIENCE_TYPE;

-------------------------------------------------------------------------
-- Function: Sets the mimimum salience threshold.
--
-- Parameters: SALIENCE - Set the salience threshold to the value of this
--                        parameter
--             Returns the old salience threshold.
-------------------------------------------------------------------------
function SET_SALIENCE_THRESHOLD(SALIENCE: in INTEGER) return INTEGER;

-------------------------------------------------------------------------
-- Function:  Returns the default limit on rule firings for RUN.
--
-------------------------------------------------------------------------
function GET_LIMIT_DEFAULT return INTEGER;

-------------------------------------------------------------------------
-- Function: Sets the default limit on rule firings for run.
--           Returns the old run default limit.
-------------------------------------------------------------------------
function SET_LIMIT_DEFAULT(LIMIT: in INTEGER) return INTEGER;
```

```
   ------------------------------------------------------------------------
   -- Function: Returns a boolean value indicating whether ART/Ada will
   --           print informational messages.  A True value indicates that
   --           the messages will be printed; False means that the messages
   --           will be suppressed.  The value of PRINT_MESSAGE_FLAG is read.
   ------------------------------------------------------------------------
   function GET_PRINT_MESSAGES return BOOLEAN;

   ------------------------------------------------------------------------
   -- Function: This procedure sets the value of PRINT_MESSAGE_FLAG, a
   --           state variable that controls the printing of messages.
   --           If PRINT_MESSAGE_FLAG is True then messages will be printed.
   --           If PRINT_MESSAGE_FLAG is False then messages will be
   --           suppressed.
   --
   -- Parameters: VALUE - The procedure sets PRINT_MESSAGE_FLAG to VALUE
   --                     (True or False)
   ------------------------------------------------------------------------
   function SET_PRINT_MESSAGES(VALUE: in BOOLEAN) return BOOLEAN;

   ------------------------------------------------------------------------
   -- Function: Complete the execution of all rhs actions of the current
   --           rule and halts the inference engine.
   --
   ------------------------------------------------------------------------
   procedure HALT;

end INFER_ENG_SUB;
```

3.4.11 Package Specification of INIT_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: J. T. Badura
--
-- Package: INIT_SUB
--
-- Function: This package contains subprograms that initialize ART.
--
-- State Variables:
--           None
-- State Variable Initialization:
--           None
-- Change Log:
--
------------------------------------------------------------------------
package INIT_SUB is
   ---------------------------------------------------------------------
   -- Function: Initialize the two major data structures:
   --           CUR_USER and CUR_APPL.
   ---------------------------------------------------------------------
   procedure INIT_GLOBALS;

   ---------------------------------------------------------------------
   -- Function: Initializes ART/Ada
   ---------------------------------------------------------------------
   procedure DO_CLEAR;

end INIT_SUB;
```

3.4.12 Package Specification of GLOBAL_DCL

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: GLOBAL_DCL
--
-- Function: This package contains global data structures for the user
--           and the application.
--
-- State Variables:
--           SYSTEM_INITIALIZED
--              TRUE => system is initialized
--              FALSE => system is NOT initialized
--
-- State Variable Initialization:
--           SYSTEM_INITIALIZED := FALSE
--
-- Change Log:
--
------------------------------------------------------------------------
with STRUCT_DCL, SYSTEM_DCL, ERROR_HDL_SUB, TEXT_IO;
use  STRUCT_DCL, SYSTEM_DCL, ERROR_HDL_SUB;
package GLOBAL_DCL is

   -- This record contains all global variables which are application specific.

   type APPLICATION is
   record
```

```
   -- these are initialized at startup
   PROMPT : STRING(1..4) := "=>  ";                -- interactive prompt is "=>  "

   -- these are initialized as needed
   FUN_LIST : ART_OBJECT;
   INCR_JOIN_NODES : JOIN_REF_PTR_TYPE;
   INCR_PAT_NODES : PAT_REF_PTR_TYPE;
   LIST_OF_RULES : RULEINFO_PTR_TYPE;
   DEFLIST : DEFFACT_PTR_TYPE;
   RELATION_TABLE : RELATION_PTR_ARRAY_TYPE(1..TABLE_SIZE);
   INTERNAL_HASHTABLE : ART_OBJECT_ARRAY_TYPE(1..INTERNAL_AO_TABLE_SIZE);

   TEST_PATTERNS : TEST_PTR_TYPE;
   VARIABLE_RELATIONS : PATTERN_NODE_PTR_TYPE;
   SALIENCE_CODE : ART_CODE_PTR_TYPE;

   EMIT_LOAD_CR      : INTEGER;
   VARS_FOUND        : BOOLEAN;
   LAST_JOIN_NODE_ID : INTEGER;
end record;

type APPLICATION_PTR_TYPE is access APPLICATION;

--  This record contains all global variables which are user specific.

type USER is
record
   -- these are initialized at startup
   AGENDA               : ACTIVATION_PTR_TYPE := null;
   ID                   : INTEGER := -1;
   NID                  : INTEGER := -1;
   RULE_FIRE_ERROR      : BOOLEAN := FALSE;
   SPC_SIG              : BOOLEAN := TRUE;
   TERMINATE_EXECUTION  : BOOLEAN := FALSE;
   ART_ARG_COUNT        : INTEGER := 0;
   FIRST_TIME           : INTEGER := 1;
   EPH_LAST_COUNT       : INTEGER := 0;
   EPH_SYMBOL_COUNT     : INTEGER := 0;
   DRIBBLE_STREAM       : TEXT_IO.FILE_TYPE;
   LAST_RETURN          : ART_OBJECT := null;
   PERM_INDEX           : INTEGER := 0;
   PERM_LENGTH          : INTEGER := 0;
   PERM_MAX             : INTEGER := 0;
   PERM_TABLE           : ART_OBJECT_ARRAY_PTR_TYPE := null;
   LOGICAL_PART_MATCH   : PART_MATCH_PTR_TYPE := null;
   GARBAGE_FACTS        : ART_OBJECT := NULL;
   RHS_RECLAIMS         : RHS_BIND_PTR_TYPE   := null;
   CURRENT_PART_MATCH   : PART_MATCH_PTR_TYPE := null;
   VAR_INDEX            : INTEGER;
   RTR_COUNT            : INTEGER := 0;
   SP                   : POSITIVE := 1;
   WATCH_ACTIVATIONS    : BOOLEAN := FALSE;
   WATCH_FACTS          : BOOLEAN := FALSE;
   WATCH_RULES          : BOOLEAN := FALSE;
   FACT_IDS_BEING_WATCHED : INTEGER_LIST_TYPE; -- NULL
   IN_PATTERN_NET       : BOOLEAN := FALSE;

   ERROR_BUFFER         : STR_PTR_TYPE;

   RESET_ALREADY_PERFORMED : BOOLEAN;

   STR_STK_SP           : INTEGER;
   BUF_PTR              : INTEGER;
   AO_BLOCK_LENGTH      : INTEGER;
```

```
    AO_SP                : INTEGER;
    AO_STACK             : ART_OBJECT_ARRAY_PTR_TYPE;

    CURRENT_FUNCTION     : FUNC_ID_TYPE;
    CURRENT_FACT         : ART_OBJECT;
    GLOBAL_HASHTABLE     : ART_OBJECT_ARRAY_PTR_TYPE;   -- GLOBAL_HASH_LENGTH
    FLOAT_HASHTABLE      : ART_OBJECT_ARRAY_PTR_TYPE;   -- TABLE_SIZE
    INT_HASHTABLE        : ART_OBJECT_ARRAY_PTR_TYPE;   -- TABLE_SIZE
    STR_HASHTABLE        : ART_OBJECT_ARRAY_PTR_TYPE;   -- TABLE_SIZE
    SYM_HASHTABLE        : ART_OBJECT_ARRAY_PTR_TYPE;   -- TABLE_SIZE
    FACT_TABLE           : ART_OBJECT_ARRAY_PTR_TYPE;   -- FACT_HASH_LENGTH
    SEQ_HASHTABLE        : ART_OBJECT_ARRAY_PTR_TYPE;   -- FACT_HASH_LENGTH
    CURRENT_RULE         : RULEINFO_PTR_TYPE;
    CURRENT_PATTERN      : PATTERN_NODE_PTR_TYPE;
    CURRENT_JOIN         : JOIN_NODE_PTR_TYPE;
    IN_DEFFACTS_P        : BOOLEAN;
    GLOB_STACK           : VALUE_SPEC_ARRAY_TYPE(1..STACK_MAX);
    STACK                : POSITIVE := 1;

    ------------------------------------------------------------------
    -- ART/Ada keeps tract of FLOAT, STRING, and ART_OBJECT seperately.
    -- GLOB_STACK will maintain the pointers to these items.
    ------------------------------------------------------------------
    FLOAT_STACK : FLOAT_ARRAY_TYPE(1..STACK_MAX);
    STR_PTR_STACK : STR_PTR_ARRAY_TYPE(1..STACK_MAX);
    ART_OBJECT_STACK : ART_OBJECT_ARRAY_TYPE(1..STACK_MAX);

    FLOAT_SP : POSITIVE := 1;
    STR_PTR_SP : POSITIVE := 1;
    ART_OBJECT_SP : POSITIVE := 1;

    FLOAT_STACK_OFFSET : NATURAL := 0;
    STR_PTR_STACK_OFFSET : NATURAL := 0;
    ART_OBJECT_STACK_OFFSET : NATURAL := 0;

    L_JOIN_TABLE         : PART_MATCH_PTR_ARRAY_PTR_TYPE;   -- JOIN_TABLE_LENGTH
    R_JOIN_TABLE         : PART_MATCH_PTR_ARRAY_PTR_TYPE;   -- JOIN_TABLE_LENGTH
    LAST_BREAKPOINT_REACHED : ACTIVATION_PTR_TYPE;
    PRINT_MESSAGES       : BOOLEAN;
    SAFE_FROM_GC         : BOOLEAN;
    DEFAULT_RUN_LIMIT    : INTEGER;
    CPU_TIME             : INTEGER;
    REAL_TIME            : INTEGER;
    SALIENCE_THRESHOLD   : INTEGER;

    LAST_RULE_BREAK_STATUS : INTEGER;

    RUN_STATE            : ERROR_LOC;
    SAVED_TOKEN          : ART_OBJECT;
    SAVED_TOKEN2         : ART_OBJECT;

    RULES_FIRED_SINCE_RESET : INTEGER;
    CUR_ASYNC_FUN        : STR_PTR_TYPE;   -- Current asynchronous function
    TEMPORARY_DATA_STATE: BOOLEAN;

-- THIS FIELD POINTS TO THE APPROPRIATE APPLICATION STRUCT
    APPL                 : APPLICATION_PTR_TYPE;
 end record;

 type USER_PTR_TYPE is access USER;

 CUR_USER : USER_PTR_TYPE;            -- current user

 CUR_APPL : APPLICATION_PTR_TYPE;     -- current application
```

```
      SYSTEM_INITIALIZED : BOOLEAN := FALSE;

   end GLOBAL_DCL;
```

3.4.13 Package Specification of SYSTEM_DCL

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: SYSTEM_DCL
--
-- Function: This package contains data structures for system variables.
--
-- State Variables:
--
-- State Variable Initialization:
--
-- Change Log:
--
------------------------------------------------------------------------
with STRUCT_DCL;
use STRUCT_DCL;
package SYSTEM_DCL is

   HASHSIZE : NATURAL := 167;

   -- initial stack size
   INITIAL_STACK_SIZE : NATURAL := 50;


   -- maximum stack size
   STACK_MAX : NATURAL := 1000;
```

```
      -- The maximum length of a symbol.
      SYMBOL_LENGTH : NATURAL := 512;

      FACT_HASH_LENGTH : NATURAL := 1009;
      TABLE_SIZE : NATURAL := 1009;
      INTERNAL_AO_TABLE_SIZE : NATURAL := 1009;
      GLOBAL_HASH_LENGTH : NATURAL := 101;
      JOIN_TABLE_LENGTH : NATURAL := 10007;
      -- The above should be prime.

      -- T and NIL
      T : ART_OBJECT;
      NIL : ART_OBJECT;

      MAX_SALIENCE: constant INTEGER :=  10_000;

      MIN_SALIENCE: constant INTEGER := -10_000;

   end SYSTEM_DCL;
```

3.4.14 Package Specification of IO_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: J. T. Badura
--
-- Package: IO_SUB
--
-- Function: This package contains subprograms that print objects out to the
--           screen or to a file.
--
-- State Variables: None
--
-- State Variable Initialization: None
--
-- Change Log:
--
------------------------------------------------------------------------
with STRUCT_DCL;
use  STRUCT_DCL;
package IO_SUB is

   ---------------------------------------------------------------------
   -- Function: Prints the object followed by a CR LF to the standard output.
   --           Standard output is, by default, the screen.
   --
   -- Parameters: OBJECT - The object to be printed
   ---------------------------------------------------------------------
   procedure PRINT(OBJECT: ART_OBJECT);
   procedure PRINT(OBJECT: INTEGER);
```

```
   procedure PRINT(OBJECT: FLOAT_TYPE);
   procedure PRINT(OBJECT: STRING);

   ------------------------------------------------------------------------
   -- Function: Writes the object out to the screen.
   --
   -- Parameters: OBJECT - The object that is transferred to a string.
   ------------------------------------------------------------------------
   function WRITE_TO_STRING(OBJECT: ART_OBJECT) return STRING;
   function WRITE_TO_STRING(OBJECT: INTEGER)    return STRING;
   function WRITE_TO_STRING(OBJECT: FLOAT_TYPE) return STRING;

   ------------------------------------------------------------------------
   -- Function: Prints the object to the standard output.
   --           Standard output is, by default, the screen.
   --
   -- Parameters: OBJECT - The object to be printed
   ------------------------------------------------------------------------
   procedure PRINC(OBJECT: ART_OBJECT);
   procedure PRINC(OBJECT: INTEGER);
   procedure PRINC(OBJECT: FLOAT_TYPE);
   procedure PRINC(OBJECT: STRING);

   ------------------------------------------------------------------------
   -- Function: Prints a CR to the standard output
   --           Standard output is, by default, the screen.
   ------------------------------------------------------------------------
   procedure TERPRI;

   ------------------------------------------------------------------------
   -- Function: Redirect output from STANDARD_OUTPUT to a file.
   ------------------------------------------------------------------------
   procedure DRIBBLE (FNAME : STRING);

   ------------------------------------------------------------------------
   -- Function: Redirect output from a file to STANDARD_OUTPUT.
   ------------------------------------------------------------------------
   procedure DRIBBLE;

   ------------------------------------------------------------------------
   -- Function: Prints out a segment of a fact to the screen
   ------------------------------------------------------------------------
   procedure PRINT_SEGMENT(SEG    : ART_OBJECT_ARRAY_PTR_TYPE;
                           LENGTH: NATURAL);

   ------------------------------------------------------------------------
   -- Function: Outputs the difference between TIME_1 and TIME_2
   ------------------------------------------------------------------------
   procedure OUTPUT_TIME(TIME_1: INTEGER;
                         TIME_2: INTEGER);

end IO_SUB;
```

3.4.15 Package Specification of PATTERN_NET_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: PATTERN_NET_SUB
--
-- Function: This package contains subprograms necessary to interpret
--           the pattern network data structure.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package PATTERN_NET_SUB is

   ------------------------------------------------
   -- PAT_ENTER: Tests a new fact against the pattern network and
   -- sends partial matches to the join network.
   ------------------------------------------------
   procedure PAT_ENTER(FACT_PTR: ART_OBJECT);
```

```
end PATTERN_NET_SUB;
```

3.4.16 Package Specification of JOIN_NET_SUB

```
------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: JOIN_NET_SUB
--
-- Function: This package contains subprograms necessary to interpret
--           the join network data structure.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package JOIN_NET_SUB is

   ------------------------------------------------
   -- R_DRIVE: Enter the join net from the right side, i.e. from the pattern
   -- network.
   ------------------------------------------------
   procedure R_DRIVE(JOIN: JOIN_NODE_PTR_TYPE;
```

```
        RHS_BIND: RHS_BIND_PTR_TYPE);

    ------------------------------------------------
    -- L_DRIVE:
    -- l_drive drives downward from the left hand join.
    -- First saving the left in the hash table,
    -- then finding and pursuing all matches.
    -- Should only be entered from itself, and from r_drive,
    -- with the possible exception of intialization
    -- of the root join.
    ------------------------------------------------
    procedure L_DRIVE(L_BINDS: RHS_BIND_PTR_ARRAY_PTR_TYPE;
                      R_BIND: RHS_BIND_PTR_TYPE;
                      JOIN: JOIN_NODE_PTR_TYPE);

     ------------------------------------------------
     -- REMOVE_RHS_BIND_MATCHES: Removes all the matches that derived from
     -- this rhs_bind and places the rhs_bind onto a list where it will be
     -- reclaimed at the end of processing.
     ------------------------------------------------
     procedure REMOVE_RHS_BIND_MATCHES(THIS_RHS_BIND: RHS_BIND_PTR_TYPE);

    ------------------------------------------------
    -- CREATE_INITIAL_DEFFACT:  Creates the initial fact, (initial-fact),
    --   and places it on the deffacts list.
    ------------------------------------------------
    procedure CREATE_INITIAL_DEFFACT;

end JOIN_NET_SUB;
```

3.4.17 Package Specification of ART_OBJECT_UTIL_SUB

```
------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: ART_OBJECT_UTIL_SUB
--
-- Function: This package contains utilities for the ART_OBJECT manipulation.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--      9-14-88 SDL - Added push_art_object, place_art_object, push_int, place_int.
--
------------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package ART_OBJECT_UTIL_SUB is

   ---------------------------------------------------------------------------
   -- VALUE_IN_ART_OBJECT_REF_LIST_P: Checks to see whether a given value is
   -- present in a list or not.
   ---------------------------------------------------------------------------
   function VALUE_IN_ART_OBJECT_REF_LIST_P(LIST : ART_OBJECT_LIST_TYPE;
```

```
                                        VALUE : ART_OBJECT) return BOOLEAN;

    ------------------------------------------------------------------------
    -- SORT_ART_OBJECT_REF: This functions sorts a list of art_object references
    -- based on lexical ordering of the schema names in the given argument list.  A
    -- simple bubble sort will be used until the need for a more advanced sort is
    -- proven.
    ------------------------------------------------------------------------
    generic
         with function LESS_THAN_FUNCTION (OBJECT_1, OBJECT_2 : ART_OBJECT)
            return BOOLEAN;
    function SORT_ART_OBJECT_REF(ART_OBJECT_REF_LIST: ART_OBJECT_LIST_TYPE)
         return ART_OBJECT_LIST_TYPE;

    ------------------------------------------------------------------------
    -- ART_OBJECT_LESS_THAN: Compares two art-objects and decides whether,
    -- according to a lexicographical ordering of my own, the first art-object is
    -- less than the first one or not. Used here in a search routine.
    ------------------------------------------------------------------------
    function ART_OBJECT_LESS_THAN (OBJECT_1, OBJECT_2 : ART_OBJECT)
         return BOOLEAN;

    ------------------------------------------------------------------------
    -- CREATE_MERGED_ART_OBJECT_REF_LIST: Nondestructively creates a merged copy
    -- of the two given lists that contains no duplicates.
    ------------------------------------------------------------------------
    function CREATE_MERGED_ART_OBJECT_REF_LIST(LIST_1, LIST_2 : ART_OBJECT_LIST_TYPE)
         return ART_OBJECT_LIST_TYPE;

    ------------------------------------------------------------------------
    -- INSERT_ART_OBJECT_REF_INTO_LIST: Inserts a reference to a value into a list,
    -- maintaining the order.
    ------------------------------------------------------------------------
    function INSERT_ART_OBJECT_REF_INTO_LIST(ART_OBJECT_LIST : ART_OBJECT_LIST_TYPE;
                                             NEW_VALUE : ART_OBJECT) return ART_OBJECT_LIST_TYPE;

    ------------------------------------------------------------------------
    -- DEALLOCATE_ART_OBJECT_REF_LIST: Deallocates a list of art_object references.
    ------------------------------------------------------------------------
    procedure DEALLOCATE_ART_OBJECT_REF_LIST(THIS_REF : ART_OBJECT_LIST_TYPE);

    ------------------------------------------------------------------------
    -- PRINT_ART_OBJECT_REF_LIST: Prints out a list of art-object references, one
    -- reference per line and indented.
    ------------------------------------------------------------------------
    procedure PRINT_ART_OBJECT_REF_LIST(REF_LIST : ART_OBJECT_LIST_TYPE;
                                        INDENTATION : INTEGER);

    ------------------------------------------------------------------------
    -- subprograms for ART_OBJECT_STACK
    ------------------------------------------------------------------------
    function CREATE_ART_OBJECT_STACK(INITIAL_SIZE : INTEGER)
          return ART_OBJECT_STACK_TYPE;

    procedure RETURN_ART_OBJECT_STACK(STACK : ART_OBJECT_STACK_TYPE);

    procedure GROW_ART_OBJECT_STACK(STACK : ART_OBJECT_STACK_TYPE);

    procedure PUSH_ART_OBJECT(THIS_STACK : ART_OBJECT_STACK_TYPE;
                              INDEX : INTEGER;
                              VALUE : ART_OBJECT);
--    pragma INLINE(PUSH_ART_OBJECT);

    procedure PLACE_ART_OBJECT(THIS_STACK : ART_OBJECT_STACK_TYPE;
```

```
                                INDEX : INTEGER;
                                VALUE : ART_OBJECT);
--    pragma INLINE(PLACE_ART_OBJECT);

   ------------------------------------------------------------------------
   -- subprograms for INTEGER_STACK
   ------------------------------------------------------------------------
   function CREATE_INT_STACK(INITIAL_SIZE : INTEGER)
        return INT_STACK_TYPE;

   procedure RETURN_INT_STACK(STACK : INT_STACK_TYPE);

   procedure GROW_INT_STACK(STACK : INT_STACK_TYPE);

   procedure PUSH_INT(THIS_STACK : INT_STACK_TYPE;
                      INDEX : INTEGER;
                      VALUE : INTEGER);
--    pragma INLINE(PUSH_INT);

   procedure PLACE_INT(THIS_STACK : INT_STACK_TYPE;
                       INDEX : INTEGER;
                       VALUE : INTEGER);
--    pragma INLINE(PUSH_INT);

end ART_OBJECT_UTIL_SUB;
```

3.4.18 Package Specification of AGENDA_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: AGENDA_SUB
--
-- Function: This package contains subprograms necessary to manage
--           the Agenda mechanism.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package AGENDA_SUB is

   ------------------------------------------------
   -- ADD_ACTIVATION: Adds a rule activation to the agenda.  This
   --   function is called when all the patterns on the LHS of a
   --   rule have been satisfied.
   ------------------------------------------------
```

```
    function ADD_ACTIVATION(JOIN  : JOIN_NODE_PTR_TYPE;
                            BINDS : PART_MATCH_PTR_TYPE)
      return ACTIVATION_PTR_TYPE;

    -------------------------------------------------
    -- DELETE_ACTIVATION: Remove an activation
    -- from the agenda.  Called when a fact
    -- gets retracted.
    -------------------------------------------------
    procedure DELETE_ACTIVATION(ACT: in out ACTIVATION_PTR_TYPE);

end AGENDA_SUB;
```

3.4.19 Package Specification of INTERPRETER_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: Interpreter_sub
--
-- Function: This package contains subprograms necessary to interpret
--           the code vectors.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
-------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package INTERPRETER_SUB is

   ----------------------------------------------------------------------
   -- Function: This function evaluates(interpretes) the code vector.
   --
   -- Parameter:
   --   ART_CODE - It is a pointer to a stack which contains code vectors.
```

```
    --   P_LEFT   - It is a left argument for pattern net eval.
    --   P_RIGHT  - It is a right argument for pattern net eval.
    --   J_LEFT   - It is a left argument for join net eval.
    --   J_RIGHT  - It is a right argument for join net eval.
    ------------------------------------------------------------------------
    function EVAL(ART_CODE : ART_CODE_PTR_TYPE;
                  P_LEFT   : ART_OBJECT                  := null;
                  P_RIGHT  : ART_OBJECT_ARRAY_PTR_TYPE   := null;
                  J_LEFT   : RHS_BIND_PTR_ARRAY_PTR_TYPE := null;
                  J_RIGHT  : RHS_BIND_PTR_TYPE           := null) return ART_OBJECT;

    ------------------------------------------------------------------------
    -- Function: This function calls Ada subprograms.
    --
    -- Parameter:
    --   FUNC_ID - It is an internal ID of the Ada programs.
    ------------------------------------------------------------------------
    procedure FUNCALL(FUNC_ID: FUNC_ID_TYPE);

end INTERPRETER_SUB;
```

3.4.20 Package Specification of ALLOC_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--             Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--             These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: ALLOC_SUB
--
-- Function: This package contains declaration of subprograms for
--           the memory allocation.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
-------------------------------------------------------------------------------

with STRUCT_DCL, UNCHECKED_DEALLOCATION;
use STRUCT_DCL;
package ALLOC_SUB is

  ------------------------------------------------------------------------
  -- Function: It allocates a PART_MATCH record.
  ------------------------------------------------------------------------
  function GET_PART_MATCH(LAST_JOIN_FLAG : BOOLEAN;
                          LENGTH     : INTEGER) return PART_MATCH_PTR_TYPE;
```

```
-------------------------------------------------------------------------
-- Function: It frees a PART_MATCH record.
-------------------------------------------------------------------------
procedure RTN_PART_MATCH(ITEM   : in out PART_MATCH_PTR_TYPE;
                         LENGTH     : INTEGER);

-------------------------------------------------------------------------
-- Function: It allocates a RHS_BIND record.
-------------------------------------------------------------------------
function GET_RHS_BIND(LENGTH     : INTEGER) return RHS_BIND_PTR_TYPE;

-------------------------------------------------------------------------
-- Function: It frees a RHS_BIND record.
-------------------------------------------------------------------------
procedure RTN_RHS_BIND(ITEM : in out RHS_BIND_PTR_TYPE;
                       LENGTH     : INTEGER);

-------------------------------------------------------------------------
-- Function: It allocates a RHS_BIND record.
-------------------------------------------------------------------------
function GET_RHS_BIND_PTR_ARRAY(LENGTH : INTEGER) return RHS_BIND_PTR_ARRAY_PTR_TYPE;

-------------------------------------------------------------------------
-- Function: It frees a RHS_BIND array.
-------------------------------------------------------------------------
procedure RTN_RHS_BIND_PTR_ARRAY is new
  UNCHECKED_DEALLOCATION(RHS_BIND_PTR_ARRAY_TYPE, RHS_BIND_PTR_ARRAY_PTR_TYPE);

-------------------------------------------------------------------------
-- Function: It creates a new fact.
-------------------------------------------------------------------------
function GET_FACT return ART_OBJECT;

-------------------------------------------------------------------------
-- Function: It frees a new fact.
-------------------------------------------------------------------------
procedure RTN_FACT is new
  UNCHECKED_DEALLOCATION(ELEMENT, ART_OBJECT);

-------------------------------------------------------------------------
-- Function: It creates a new TEMPLATE.
-------------------------------------------------------------------------
function GET_TEMPLATE(LENGTH : INTEGER) return TEMPLATE_TYPE;

-------------------------------------------------------------------------
-- Function: It frees a new TEMPLATE.
-------------------------------------------------------------------------
procedure RTN_TEMPLATE is new
  UNCHECKED_DEALLOCATION(SEQUENCE_TYPE, SEQUENCE_PTR_TYPE);

-------------------------------------------------------------------------
-- Function: It creates a new deffact.
-------------------------------------------------------------------------
function GET_DEFFACT return DEFFACT_PTR_TYPE;

-------------------------------------------------------------------------
-- Function: It frees a new deffact.
-------------------------------------------------------------------------
procedure RTN_DEFFACT is new
  UNCHECKED_DEALLOCATION(DEFFACT, DEFFACT_PTR_TYPE);

-------------------------------------------------------------------------
-- Function: It creates a new activation.
-------------------------------------------------------------------------
```

```
   function GET_ACTIVATION return ACTIVATION_PTR_TYPE;

   ------------------------------------------------------------------------
   -- Function: It frees a new activation.
   ------------------------------------------------------------------------
   procedure RTN_ACTIVATION is new
     UNCHECKED_DEALLOCATION(ACTIVATION, ACTIVATION_PTR_TYPE);

   ------------------------------------------------------------------------
   -- Function: It creates a new join_ref.
   ------------------------------------------------------------------------
   function GET_JOIN_REF return JOIN_REF_PTR_TYPE;

   ------------------------------------------------------------------------
   -- Function: It frees a new join_ref.
   ------------------------------------------------------------------------
   procedure RTN_JOIN_REF is new
     UNCHECKED_DEALLOCATION(JOIN_REF, JOIN_REF_PTR_TYPE);

   ------------------------------------------------------------------------
   -- Function: It creates a LOG_SUPPORT record.
   ------------------------------------------------------------------------
   function GET_LOG_SUPPORT return LOG_SUPPORT_PTR_TYPE;

   ------------------------------------------------------------------------
   -- Function: It frees a LOG_SUPPORT record.
   ------------------------------------------------------------------------
   procedure RTN_LOG_SUPPORT is new
     UNCHECKED_DEALLOCATION(LOG_SUPPORT, LOG_SUPPORT_PTR_TYPE);

   ------------------------------------------------------------------------
   -- Function: It frees a sequence.
   ------------------------------------------------------------------------
   procedure FREE_SEQUENCE is new
     UNCHECKED_DEALLOCATION(ELEMENT, ART_OBJECT);

   function GET_AO_STACK(LENGTH : INTEGER) return ART_OBJECT_ARRAY_PTR_TYPE;

   procedure RTN_AO_STACK is new
     UNCHECKED_DEALLOCATION(ART_OBJECT_ARRAY_TYPE, ART_OBJECT_ARRAY_PTR_TYPE);

   function GET_SYMBOL return ART_OBJECT;

   ------------------------------------------------------------------------
   -- Function: It returns a pointer to a string of length L
   ------------------------------------------------------------------------
   function GET_STRING(L: NATURAL) return STR_PTR_TYPE;

   function GET_INT_RECORD return ART_OBJECT;

   function GET_FLOAT_RECORD return ART_OBJECT;

end ALLOC_SUB;
```

3.4.21 Package Specification of MACRO_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: MACRO_SUB
--
-- Function: This package contains subprograms that are equivalent to
--           C macros in ART-IM.  These subprograms are used with
--           pragma INLINE.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------

with STRUCT_DCL, UNCHECKED_CONVERSION;
use STRUCT_DCL;
package MACRO_SUB is

   ------------------------------------------------------------------
   -- Function: It returns OBJ.DATA.FACT.ID.
   ------------------------------------------------------------------
   function FACT_ID (OBJ : ART_OBJECT) return NATURAL;
```

```
pragma INLINE(FACT_ID);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.FACT.ID to ID.
-------------------------------------------------------------------
procedure SET_FACT_ID (OBJ : ART_OBJECT;
                       ID : NATURAL);
pragma INLINE(SET_FACT_ID);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.FACT.BIND_LIST.
-------------------------------------------------------------------
function FACT_BIND_LIST (OBJ : ART_OBJECT) return RHS_BIND_PTR_TYPE;
pragma INLINE(FACT_BIND_LIST);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.FACT.BIND_LIST to BIND_LIST.
-------------------------------------------------------------------
procedure SET_FACT_BIND_LIST (OBJ : ART_OBJECT;
                              BIND_LIST : RHS_BIND_PTR_TYPE);
pragma INLINE(SET_FACT_BIND_LIST);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.FACT.SUPPORT.
-------------------------------------------------------------------
function FACT_SUPPORT (OBJ : ART_OBJECT) return LOG_SUPPORT_PTR_TYPE;
pragma INLINE(FACT_SUPPORT);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.FACT.SUPPORT to SUPPORT.
-------------------------------------------------------------------
procedure SET_FACT_SUPPORT (OBJ : ART_OBJECT;
                            SUPPORT : LOG_SUPPORT_PTR_TYPE);
pragma INLINE(SET_FACT_SUPPORT);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.FACT.PROPOSITION.
-------------------------------------------------------------------
function FACT_PROPOSITION (OBJ : ART_OBJECT) return ART_OBJECT;
pragma INLINE(FACT_PROPOSITION);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.FACT.PROPOSITION to PROPOSITION.
-------------------------------------------------------------------
procedure SET_FACT_PROPOSITION (OBJ : ART_OBJECT;
                                PROPOSITION : ART_OBJECT);
pragma INLINE(SET_FACT_PROPOSITION);

-------------------------------------------------------------------
-- Function: It returns
--     OBJ.DATA.FACT.PROPOSITION.DATA.SEQ.LENGTH.
-------------------------------------------------------------------
function FACT_LENGTH (OBJ : ART_OBJECT) return NATURAL;
pragma INLINE(FACT_LENGTH);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.FACT.PROPOSITION.DATA.SEQ.LENGTH
--           to LENGTH.
-------------------------------------------------------------------
procedure SET_FACT_LENGTH (OBJ : ART_OBJECT;
                           LENGTH : NATURAL);
pragma INLINE(SET_FACT_LENGTH);

-------------------------------------------------------------------
```

```
-- Function: It return
--    OBJ.DATA.FACT.PROPOSITION.DATA.SEQ.DATA.
-------------------------------------------------------------------
function FACT_ATOMS (OBJ : ART_OBJECT) return ART_OBJECT_ARRAY_PTR_TYPE;
pragma INLINE(FACT_ATOMS);

-------------------------------------------------------------------
-- Function: It sets
--   OBJ.DATA.FACT.PROPOSITION.DATA.SEQ.DATA
--   to ATOMS.
-------------------------------------------------------------------
procedure SET_FACT_ATOMS (OBJ : ART_OBJECT;
                          ATOMS : ART_OBJECT_ARRAY_PTR_TYPE);
pragma INLINE(SET_FACT_ATOMS);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.SEQ.LENGTH.
-------------------------------------------------------------------
function SEQ_LENGTH (OBJ : ART_OBJECT) return NATURAL;
pragma INLINE(SEQ_LENGTH);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.SEQ.LENGTH to LENGTH.
-------------------------------------------------------------------
procedure SET_SEQ_LENGTH (OBJ : ART_OBJECT;
                          LENGTH : NATURAL);
pragma INLINE(SET_SEQ_LENGTH);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.SEQ.DATA.
-------------------------------------------------------------------
function SEQ_ELT (OBJ : ART_OBJECT) return ART_OBJECT_ARRAY_PTR_TYPE;
pragma INLINE(SEQ_ELT);

-------------------------------------------------------------------
-- Function: It sets OBJ.DATA.SEQ.DATA to ELT.
-------------------------------------------------------------------
procedure SET_SEQ_ELT (OBJ : ART_OBJECT;
                       ELT : ART_OBJECT_ARRAY_PTR_TYPE);
pragma INLINE(SET_SEQ_ELT);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.SEQ.REF_COUNT.
-------------------------------------------------------------------
function SEQ_REF_COUNT (OBJ : ART_OBJECT) return INTEGER;
pragma INLINE(SEQ_REF_COUNT);

-------------------------------------------------------------------
-- Function: It returns OBJ.DATA.SEQ.REF_COUNT to REF_COUNT.
-------------------------------------------------------------------
procedure SET_SEQ_REF_COUNT (OBJ : ART_OBJECT;
                             REF_COUNT : INTEGER);
pragma INLINE(SET_SEQ_REF_COUNT);

-------------------------------------------------------------------
-- This function increments an integer by one.
-- This is equivalent to ++ in C.
-------------------------------------------------------------------
procedure INCREMENT( NUM : in out INTEGER );
pragma INLINE(INCREMENT);

-------------------------------------------------------------------
-- This function decrements an integer by one.
-- This is equivalent to -- in C.
```

```
    ------------------------------------------------------------------
    procedure DECREMENT( NUM : in out INTEGER );
    pragma INLINE(DECREMENT);

    ------------------------------------------------------------------
    -- This function set MARKED_FLAG of ART_OBJECT to TRUE;
    ------------------------------------------------------------------
    procedure MARK_AO( OBJ : ART_OBJECT );
    pragma INLINE(MARK_AO);

    ------------------------------------------------------------------
    -- This function converts ART_OBJECT to INTEGER.
    ------------------------------------------------------------------
    function CONVERT_ART_OBJECT_TO_INTEGER is new UNCHECKED_CONVERSION(ART_OBJECT, INTEGER);

    procedure NINT_PUSH_AO_FRAME(OLD_AO_SP : out INTEGER);
    pragma INLINE(NINT_PUSH_AO_FRAME);

    procedure NINT_POP_AO_FRAME(OLD_AO_SP : INTEGER);
    pragma INLINE(NINT_POP_AO_FRAME);

    function GET_PAT_VAR_VECTOR(DATA    : RHS_BIND_PTR_TYPE;
                                VAR_NUM : INDEX_TYPE) return ART_OBJECT;
    pragma INLINE(GET_PAT_VAR_VECTOR);

    function GET_JOIN_VAR( DATA : RHS_BIND_PTR_ARRAY_PTR_TYPE;
                           PAT  : INDEX_TYPE;
                           VAR_NUM : INDEX_TYPE) return ART_OBJECT;
    pragma INLINE(GET_JOIN_VAR);

end MACRO_SUB;
```

3.4.22 Package Specification of UI_INTERNAL_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: DATABASE_SUB
--
-- Function: This package contains subprograms necessary to manage
--           the ART/Ada database.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package UI_INTERNAL_SUB is

  ---------------------------------------------------------------------
  -- Function: It prints out "watch facts" messages for assert.
  ---------------------------------------------------------------------
  procedure DEBUG_FACT_ASSERT(FACT : ART_OBJECT);

  ---------------------------------------------------------------------
  -- Function: It prints out "watch activations" messages for add activation.
```

```
   ------------------------------------------------------------------
   procedure DEBUG_ACTIVATION_CREATED( ACT : ACTIVATION_PTR_TYPE );

   ------------------------------------------------------------------
   -- Function: It prints out "watch activations" messages for delete activation.
   ------------------------------------------------------------------
   procedure DEBUG_ACTIVATION_DELETED( ACT : ACTIVATION_PTR_TYPE );

   ------------------------------------------------------------------
   -- Function: It prints out "watch rules" messages for add activation.
   ------------------------------------------------------------------
   procedure DEBUG_RULE_ACTIVATED( RULENAME : ART_OBJECT);

   ------------------------------------------------------------------
   -- Function: It prints out "watch facts" messages for retract.
   ------------------------------------------------------------------
   procedure DEBUG_FACT_RETRACT(FACT : ART_OBJECT);

   ------------------------------------------------
   -- PRINT_MATCH: Prints out a list of data items that
   -- were associated with a partial match or rule
   -- instantiation.
   ------------------------------------------------
   procedure PRINT_MATCH (BINDS : PART_MATCH_PTR_TYPE);

   ------------------------------------------------
   -- PRINT_DATA_BASIS: Prints out a list of data items that
   -- were associated with a partial match or rule
   -- instantiation.
   ------------------------------------------------
   procedure PRINT_DATA_BASIS(BINDS: PART_MATCH_PTR_TYPE);

   ------------------------------------------------
   -- SHOW_DATA_ITEM:  Displays a single data item.
   ------------------------------------------------
   procedure SHOW_DATA_ITEM(DATA: ART_OBJECT);

end UI_INTERNAL_SUB;
```

3.4.23 Package Specification of MATH_SUB

```
------------------------------------------------------------------------
--
--                  C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: MATH_SUB
--
-- Function: This package contains subprograms for the math package.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
-------------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;

package MATH_SUB is

  function ART_PLUS (X : ART_OBJECT;
                     Y : ART_OBJECT) return ART_OBJECT;

  function ART_MINUS (X : ART_OBJECT;
                      Y : ART_OBJECT) return ART_OBJECT;
```

```
    function ART_TIMES (X : ART_OBJECT;
                        Y : ART_OBJECT) return ART_OBJECT;

    function ART_DIVIDE (X : ART_OBJECT;
                         Y : ART_OBJECT) return ART_OBJECT;

    function ART_NUMERIC_EQUAL (X : ART_OBJECT;
                                Y : ART_OBJECT) return BOOLEAN;

    function ART_NUMERIC_NOT_EQUAL (X : ART_OBJECT;
                                    Y : ART_OBJECT) return BOOLEAN;

    function ART_GREATER (X : ART_OBJECT;
                          Y : ART_OBJECT) return BOOLEAN;

    function ART_LESS (X : ART_OBJECT;
                       Y : ART_OBJECT) return BOOLEAN;

    function ART_GREATER_EQUAL (X : ART_OBJECT;
                                Y : ART_OBJECT) return BOOLEAN;

    function ART_LESS_EQUAL (X : ART_OBJECT;
                             Y : ART_OBJECT) return BOOLEAN;

end MATH_SUB;
```

3.4.24 Package Specification of GC_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: GC_SUB
--
-- Function: This package contains subprograms for garbage collection.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
--   Author     Date        Change
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package GC_SUB is

   ------------------------------------------------
   -- MARK_SEQUENCE: Marks a sequence for garbage collection, and all of its
   -- component data items.
   ------------------------------------------------
```

```
        procedure MARK_SEQUENCE(SEQ: ART_OBJECT);

        ------------------------------------------------------------------------------
        -- MARK_RHS_BINDS: Marks all of the data items contained in this list of RHS binds for
        -- the garbage collector.
        ------------------------------------------------------------------------------
        procedure MARK_RHS_BINDS(BINDS: RHS_BIND_PTR_TYPE);

        ---------------------------------------
        -- MARK_FACTS:
        ---------------------------------------
        procedure MARK_FACTS;

        procedure SAFE_MARK_AO( OBJECT : ART_OBJECT);

        procedure SAVE_AO(OBJECT: ART_OBJECT);

        procedure AO_SAVE_ESCAPE(OBJECT: ART_OBJECT);

    end GC_SUB;
```

3.4.25 Package Specification of COMPILER_DCL

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: COMPILER_SUB
--
-- Function: This package contains data structures and subprograms
--           specific to a particular compiler.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------
package COMPILER_SUB is

   -- VAX Ada supports double float as LONG_FLOAT.
   -- VADS supports double float as FLOAT.
   subtype INTERNAL_FLOAT_TYPE is LONG_FLOAT;

   function GET_REAL_TIME return INTEGER;

   function GET_CPU_TIME return INTEGER;
```

```
end COMPILER_SUB;
```

3.4.26 Package Specification of LOGICAL_SUB

```
------------------------------------------------------------------------
--
--                C O P Y R I G H T   N O T I C E
--
--      1) COPYRIGHT (C) 1988
--      INFERENCE CORPORATION,
--      5300 W. Century Blvd.,
--      Los Angeles, California 90045.
--      AN UNPUBLISHED WORK - - ALL RIGHTS RESERVED.
--
--      2) Restricted Rights Notice (Short Form) (April 1984)
--
--              Use, reproduction, or disclosure is
--      subject to restrictions set forth in
--      Government Cooperative Agreement Number NCC-
--      9-16 between the National Aeronautics and
--      Space Administration and the University of
--      Houston-Clear Lake and a subcontract
--      thereunder, Number 015 between the University
--      of Houston-Clear Lake and Inference
--      Corporation.
--
--      3) Restricted Rights Notice (ART/Ada)
--
--              These data constitute Inference
--      Corporation trade secrets and/or information
--      that is commercial or financial and
--      confidential or privileged.  They are
--      submitted to the Government under NASA
--      Cooperative Agreement NCC-9-16 with the
--      University of Houston-Clear Lake Research
--      Institute for Computing and Information
--      Systems (RICIS) with the understanding that
--      they will not, without the permission of
--      Inference Corporation, be used or disclosed
--      for other than evaluation purposes.
--
------------------------------------------------------------------------
-- Author: S. Daniel Lee
--
-- Package: LOGICAL_SUB
--
-- Function: This package contains subprograms necessary for
--           the logical dependency.
--
-- State Variables:
--           None
--
-- State Variable Initialization:
--           None
--
-- Change Log:
--
------------------------------------------------------------------------

with STRUCT_DCL;
use STRUCT_DCL;
package LOGICAL_SUB is

  ------------------------------------------------------------------------
  -- ADD_FACT_SUPPORT: Adds logical support information for a fact to both the
  -- fact and to the current partial match.
  ------------------------------------------------------------------------
  procedure ADD_FACT_SUPPORT(FACT : ART_OBJECT;
```

```
                                SUP_PTR : in out LOG_SUPPORT_PTR_TYPE);

    ------------------------------------------------------------------------
    -- RETRACT_SUPPORT: Retracts the support of a given partial match from all
    -- data items that it had supported before.
    ------------------------------------------------------------------------
    procedure RETRACT_SUPPORT( PART_MATCH : PART_MATCH_PTR_TYPE );

    ------------------------------------------------------------------------
    -- FREE_LOG_SUPPORT: Removes all connection from a data item to supporting
    -- partial matches. This occurs when the data item receives state support or
    -- when the item is retracted.
    ------------------------------------------------------------------------
    procedure FREE_LOG_SUPPORT( FIRST_DATA : ART_OBJECT;
                                SLOT_NAME  : ART_OBJECT;
                                VALUE      : ART_OBJECT;
                                SUP_PTR    : in out LOG_SUPPORT_PTR_TYPE);

    ------------------------------------------------------------------------
    -- VALIDATE_DATA: Validates that a piece of data (fact, schema, slot or slot
    -- value) is supported by some object that is not involved in a circle.
    ------------------------------------------------------------------------
    function VALIDATE_DATA( DATA_ITEM : ART_OBJECT;
                            SUPPORT   : LOG_SUPPORT_PTR_TYPE;
                            VALIDATING_DATA : ART_OBJECT_LIST_TYPE ) return BOOLEAN;

    ------------------------------------------------------------------------
    -- COMPUTE_LOGICAL_PART_MATCH: Computes the logical partial match.
    ------------------------------------------------------------------------
    function COMPUTE_LOGICAL_PART_MATCH(THIS_LOG_JOIN: JOIN_NODE_PTR_TYPE)
      return PART_MATCH_PTR_TYPE;

end LOGICAL_SUB;
```

3.5 Package Specification of the Booch Components

3.5.1 List_Single_Unbounded_Managed

```
-------------------------------------------------------------------------
-- This is a generic package for a singly-linked list from the Booch reusable
-- component library.
--
--   Unbounded - Denotes that the size of the object is not static.
--   Managed   - Garbage collection is provided by the component itself.
-------------------------------------------------------------------------
generic
   type Item is private;
package List_Single_Unbounded_Managed is

   type List is private;

   Null_List : constant List;

   --
   -- Copy the items from one list to another list.
   --
   procedure Copy      (From_The_List : in     List;
                        To_The_List   : in out List);

   --
   -- Remove all the items (if any) from the list and make the list null.
   --
   procedure Clear     (The_List      : in out List);

   --
   -- Add an item to the head of a list.
   --
   procedure Construct (The_Item      : in     Item;
                        And_The_List  : in out List);

   --
   -- Set the value of the head of a list to the given item.
   --
   procedure Set_Head  (Of_The_List   : in out List;
                        To_The_Item   : in     Item);
   --
   -- Exchange the tail of one list with another entire list.
   --
   procedure Swap_Tail (Of_The_List   : in out List;
                        And_The_List  : in out List);

   --
   -- Return True if the two lists have the same state.
   --
   function Is_Equal  (Left      : in List;
                       Right     : in List) return Boolean;

   --
   -- Return the current number of items in the list.
   --
   function Length_Of (The_List : in List) return Natural;

   --
   -- Return True if the list is a sequence of zero items.
   --
   function Is_Null   (The_List : in List) return Boolean;

   --
   -- Return the first item from the sequence of items in a given list.
   --
   function Head_Of   (The_List : in List) return Item;
```

```
        --
        -- Return the list denoting the tail of a given list.
        --
        function Tail_Of   (The_List : in List) return List;

        --
        -- The list cannot grow large enough to complete the desired operation.
        --
        Overflow     : exception;

        --
        -- The desired operation cannot be completed because the list already
        -- Is_Null
        --
        List_Is_Null : exception;

    private
        type Node;
        type List is access Node;
        Null_List : constant List := null;
    end List_Single_Unbounded_Managed;
```

3.5.2 String_Sequential_Unbounded_Managed_Iterator

```
-------------------------------------------------------------------------
-- This is a generic package for a string from the Booch reusable
-- component library.
--
--   Sequential - The semantics of an object are preserved only in the
--                presence of one thread of control.
--   Unbounded - Denotes that the size of the object is not static.
--   Managed   - Garbage collection is provided by the component itself.
--   Iterator  - An iterator is provided for this object.
-------------------------------------------------------------------------
generic
    type Item is private;
    type Substring is array(Positive range <>) of Item;
    with function "<" (Left  : in Item;
                       Right : in Item) return Boolean;
package String_Sequential_Unbounded_Managed_Iterator is

    type String is limited private;

    --
    -- Copy the items from one string to another string
    --
    procedure Copy     (From_The_String    : in     String;
                        To_The_String      : in out String);

    --
    -- Copy the items from one substring to another string
    --
    procedure Copy     (From_The_Substring : in     Substring;
                        To_The_String      : in out String);
    --
    --
    procedure Clear    (The_String        : in out String);

    --
    -- Catenate a string to the front of another string
    --
    procedure Prepend  (The_String        : in     String;
                        To_The_String      : in out String);

    --
    -- Catenate a substring to the front of another string
    --
    procedure Prepend  (The_Substring     : in     Substring;
                        To_The_String      : in out String);

    --
    -- Catenate a string to the end of another string
    --
    procedure Append   (The_String        : in     String;
                        To_The_String      : in out String);

    --
    -- Catenate a substring to the end of another string
    --
    procedure Append   (The_Substring     : in     Substring;
                        To_The_String      : in out String);

    --
    -- Add a string starting at the given position in another string
    --
    procedure Insert   (The_String        : in     String;
                        In_The_String      : in out String;
                        At_The_Position    : in     Positive);
```

```
--
-- Add a substring starting at the given position in another string
--
procedure Insert   (The_Substring       : in     Substring;
                    In_The_String       : in out String;
                    At_The_Position     : in     Positive);

--
-- Remove the items of a string from the given starting position to a
-- stop position, inclusive
--
procedure Delete   (In_The_String       : in out String;
                    From_The_Position   : in     Positive;
                    To_The_Position     : in     Positive);

--
-- Starting at a given position, replace the items of a string with
-- another string.
--
procedure Replace  (In_The_String       : in out String;
                    At_The_Position     : in     Positive;
                    With_The_String     : in     String);

--
-- Starting at a given position, replace the items of a string with
-- another substring
--
procedure Replace  (In_The_String       : in out String;
                    At_The_Position     : in     Positive;
                    With_The_Substring  : in     Substring);

--
-- Set the value of the given position of a string to the given item
--
procedure Set_Item (In_The_String       : in out String;
                    At_The_Position     : in     Positive;
                    With_The_Item       : in     Item);

--
-- Return True if the two strings have the same state
--
function Is_Equal        (Left              : in String;
                          Right             : in String)    return Boolean;

--
-- Return True if a substring and a string have the same state
--
function Is_Equal        (Left              : in Substring;
                          Right             : in String)    return Boolean;

--
-- Return true if a string and a substring have the same state
--
function Is_Equal        (Left              : in String;
                          Right             : in Substring) return Boolean;

--
-- Return True if the first string is lexicographically smaller than
-- the second string.
--
function Is_Less_Than    (Left              : in String;
                          Right             : in String)    return Boolean;

--
```

```
-- Return True if the first substring is lexicographically smaller than
-- the second string
--
function Is_Less_Than    (Left              : in Substring;
                          Right             : in String)    return Boolean;

--
-- Return True if the first string is lexicographically smaller than
-- the second substring
--
function Is_Less_Than    (Left              : in String;
                          Right             : in Substring) return Boolean;

--
-- Return True if the first string is lexicographically larger than
-- the second string
--
function Is_Greater_Than (Left              : in String;
                          Right             : in String)    return Boolean;

--
-- Return True if the first substring is lexicographically larger than
-- the second string
--
function Is_Greater_Than (Left              : in Substring;
                          Right             : in String)    return Boolean;

--
-- Return True if the first string is lexicographically larger than
-- the second substring
--
function Is_Greater_Than (Left              : in String;
                          Right             : in Substring) return Boolean;

--
-- Return the current number of items in the string
--
function Length_Of       (The_String        : in String)    return Natural;

--
-- Return True if the string has a zero length
--
function Is_Null         (The_String        : in String)    return Boolean;

--
-- Return the item at the given position in the string
--
function Item_Of         (The_String        : in String;
                          At_The_Position   : in Positive) return Item;

--
-- Return the substring consisting of all the items in the given string
--
function Substring_Of    (The_String        : in String)  return Substring;

--
-- Return the substring of a string from a starting position to an ending
-- position
--
function Substring_Of    (The_String        : in String;
                          From_The_Position : in Positive;
                          To_The_Position   : in Positive)return Substring;

--
```

```
    -- Visit every item in the string in order from the first position to
    -- the last position.
    --
    generic
        with procedure Process (The_Item : in  Item;
                                Continue : out Boolean);
    procedure Iterate (Over_The_String : in String);

    Overflow       : exception;  -- The string cannot grow large enough to
                                 -- complete the desired operation

    Position_Error : exception;  -- The given position is not valid for the
                                 -- string

private
    type Structure is access Substring;
    type String is
        record
            The_Length : Natural := 0;
            The_Items  : Structure;
        end record;
end String_Sequential_Unbounded_Managed_Iterator;
```

3.5.3 Floating_Point_Utilities

```
generic
    type Number is digits <>;
package Floating_Point_Utilities is

    type Base is range 2 .. 16;

    type Numbers is array(Positive range <>) of Number;

    function Integer_Part (The_Number    : in Number)     return Integer;
    function Real_Part    (The_Number    : in Number)     return Number;
    function Floor        (The_Number    : in Number)     return Integer;
    function Ceiling      (The_Number    : in Number)     return Integer;
    function Min          (Left          : in Number;
                           Right         : in Number)     return Number;
    function Min          (The_Numbers   : in Numbers)    return Number;
    function Max          (Left          : in Number;
                           Right         : in Number)     return Number;
    function Max          (The_Numbers   : in Numbers)    return Number;
    function Is_Positive  (The_Number    : in Number)     return Boolean;
    function Is_Natural   (The_Number    : in Number)     return Boolean;
    function Is_Negative  (The_Number    : in Number)     return Boolean;
    function Is_Zero      (The_Number    : in Number)     return Boolean;
    function Image_Of     (The_Number    : in Number;
                           With_The_Base : in Base := 10) return String;
    function Value_Of     (The_Image     : in String;
                           With_The_Base : in Base := 10) return Number;
    function Is_Equal     (Left          : in Number;
                           Right         : in Number)     return Boolean;

    Lexical_Error : exception;

end Floating_Point_Utilities;
```

4. Ada Call-In and Call-Out Specification for ART/Ada and ART-IM 1.5

4.1 Introduction

This section describes a portable call-in/call-out interface specification for ART/Ada and ART-IM 1.5, for communication between Ada and ART. The standard call-in/call-out mechanism will enhance portability of an ART application between ART-IM and ART/Ada.

4.2 Interface Types

The following types may be passed between ART and Ada:

:INTEGER (INTEGER) This type is an integer in Ada and an integer in ART (INTEGER in ART/Ada and long in ART-IM). The exact range of integers supported in ART varies between implementations.

:BOOLEAN (BOOLEAN) In ART, this type is either NIL or non-NIL. In Ada, this type is BOOLEAN which is TRUE or FALSE. When translating from Ada to ART TRUE will translate to T.

:FLOAT (FLOAT_TYPE) In Ada, this type is FLOAT_TYPE which is double precision float whenever the Ada compiler supports it. In ART, the exact range and precision supported varies between implementations. For ART/Ada this will be a float. For ART-IM this will be a C double.

:STRING (STR_PTR_TYPE) In Ada, this type is represented as a STR_PTR_TYPE which is a limited private type. In ART, this type is represented as an ART string. ART may or may not copy the string being passed by this mechanism when passing a string from ART to Ada. Thus it is an error to destructively modify a string passed with this mechanism. ART is responsible for freeing any space necessary for the string after exiting the current scope. The actual implementation will be based upon constraints of the underlying architecture. When transferring a string from Ada to ART, ART will always copy the string, allowing the Ada programmer to free the string at his leisure.

:SYMBOL (STR_PTR_TYPE) In Ada, this type is represented as an STR_PTR_TYPE. In ART, this type is a symbol. Case is preserved when interning an STR_PTR_TYPE as an ART symbol, just as case is preserved when passing a string to the Lisp function INTERN.

:ART-OBJECT (ART_OBJECT) This type is any ART type in ART. It is represented as a pointer to a discriminant record in ART/Ada. For ART-IM, it is an integer type which represents a C pointer to a C structure art_object. A set of Ada functions is provided to operate on these ART objects from Ada.

4.3 Scope of Objects

This section gives a detailed description of the scope of objects communicated from ART to Ada and objects communicated from Ada to ART. In both cases the prime motivation for scoping is that the caller should free all objects it allocates, (thus it should not allocate objects which it intends that the callee free). Additionally, the callee should not destructively modify objects which it did not allocate.

All objects that are not immediate fall under these constraints. For example, strings and art-objects passed from ART to Ada conform to the following semantics.

When an `art_object` or string is passed from ART by call out to an Ada function, the object is automatically reclaimed when the Ada function returns to ART. At this point, the Ada `art_object` data structure is no longer valid for use in Ada code. It is an error to retain a pointer to an automatically reclaimed `art_object` in Ada once the Ada call has returned.

When an `art_object` or string is returned from ART to Ada, it is automatically reclaimed when control returns to ART from Ada In those implementations where Ada can start up and call ART as a subroutine so that a returned value may never be reclaimed, the returned `art_object` is allocated permanently and must be freed using a freeing function supplied in Ada.

A function is supplied in Ada that accepts an `art_object` as argument and returns a permanent copy of that `art_object`. This object must be explicitly freed when no longer useful.

4.4 Call-Out from ART to Ada

The following is a grammar for def-user-fun which will be used to call out to Ada from ART:

```
(def-user-fun <fun-name> {<comment>}
        <function-spec>*)

<function-spec> ::=
        :compiler <compiler-name>      |
        :returns <return-data-type>    |
        :epname <link-editor-symbol>   |
        :args (<arg-spec>*)

<fun-name> ::= <art-symbol>

<comment>  ::= <art-string>

<compiler-name> ::=
        :VERDIX-ADA
        :DEC-ADA
        :ALSYS-ADA
```

```
<internal-data-type> ::=
        :SYMBOL
        :STRING
        :FLOAT
        :INTEGER
        :BOOLEAN

<return-data-type> ::=
        :VOID | <internal-data-type>

<link-editor-symbol> ::=
        <art-symbol> | <art-string>

<arg-spec> ::=
        (<name> <internal-data-type> <arg-attribute>*) |
        (<internal-data-type> <arg-attribute>*) |
        <internal-data-type>

<name>  ::= <art-symbol>

<arg-attribute> ::=
        <convention>    |
        <status>

<convention> ::=
        :OBJECT-POINTER |
        :VALUE-POINTER  |
        :VALUE

<status> ::=
        <optional>      |
        <rest>

<optional>  ::=
        :optional       |
        (:optional <default>)

<default>  ::= art-object

<rest> ::= :rest                ; Must be the last arg
```

4.5 Call-In from Ada to ART

The specification of the ART package will serve as the standard interface specification to call in from Ada to ART. The ART package of ART/Ada is the public package for the ART/Ada users to call in from Ada to ART/Ada.

The Ada binding for ART-IM 1.5 using the Ada pragma interface based on the standard call-in mechanism is also available separately. It can be used to call in from Ada to ART-IM 1.5. The Ada binding is composed of the specification and the body of the Ada package, *ART*. The specification of the ART package.

References

1. Booch, G. *Software Components With Ada*. Benjamin/Cummings Publishing, 1987.

2. Inference Corporation. *ART Version 3.1 Reference Manual*. Inference Corporation, 1987.

3. Inference Corporation. *ART-IM 1.0 Reference Manual*. Inference Corporation, 1988.