Approximate Model Composition for Explanation Generation

Elias Biris

Doctor of Philosophy
School of Informatics
University of Edinburgh

2003

Abstract

This thesis presents a framework for the formulation of knowledge models to support the generation of explanations for engineering systems that are represented by the resulting models. Such models are automatically assembled from instantiated generic component descriptions, known as *model fragments*. The model fragments are of sufficient detail that generally satisfies the requirements of information content as identified by the user asking for explanations.

Through a combination of fuzzy logic based evidence preparation, which exploits the history of prior user preferences, and an approximate reasoning inference engine, with a Bayesian evidence propagation mechanism, different uncertainty sources can be handled. Model fragments, each representing structural or behavioural aspects of a component of the domain system of interest, are organised in a library. Those fragments that represent the same domain system component, albeit with different representation detail, form parts of the same assumption class in the library. Selected fragments are assembled to form an overall system model, prior to extraction of any textual information upon which to base the explanations. The thesis proposes and examines the techniques that support the fragment selection mechanism and the assembly of these fragments into models.

In particular, a Bayesian network-based model fragment selection mechanism is described that forms the core of the work. The network structure is manually determined prior to any inference, based on schematic information regarding the connectivity of the components present in the domain system under consideration. The elicitation of network probabilities, on the other hand is completely automated using probability elicitation heuristics. These heuristics aim to provide the information required to select fragments which are maximally compatible with the given evidence of the fragments preferred by the user. Given such initial evidence, an existing evidence propagation algorithm is employed. The preparation of the evidence for the selection of certain fragments, based on user preference, is performed by a fuzzy reasoning evidence fabrication engine. This engine uses a set of fuzzy rules and standard fuzzy reasoning mechanisms, attempting to guess the information needs of the user and suggesting the selection of fragments of sufficient detail to satisfy such needs. Once the evidence is propagated, a single fragment is selected for each of the domain system components and hence, the final model of the entire system is constructed. Finally, a highly configurable XML-based mechanism is employed to extract explanation content from the newly formulated model and to structure the explanatory sentences for the final explanation that will be communicated to the user.

The framework is illustratively applied to a number of domain systems and is compared qualitatively to existing compositional modelling methodologies. A further empirical assessment of the performance of the evidence propagation algorithm is carried out to determine its performance limits. Performance is measured against the number of fragments that represent each of the components of a large domain system, and the amount of connectivity permitted in the Bayesian network between the nodes that stand for the selection or rejection of these fragments. Based on this assessment recommendations are made as to how the framework may be optimised to cope with real world applications.

Acknowledgements

This thesis has not been an entirely joyous journey. It started as a dream, grew to become an almost unrestrained proposal and ended as a final reality with some compromises, battling against the numerous open issues from the trenches of working a full time day job. However difficult, deep down I always felt the need to complete what I had started. In so doing, facing the difficulties was made possible through the technical contributions, effort and support from a number of people whom I now wish to give heartily thanks to:

My supervisor *Dr Qiang Shen*, who patiently and methodically tried to educate me and endured my undoubtedly soporific draft texts for paper submissions and the final thesis, whilst putting up with my belatedness and missed deadlines, due to the requirements of my day job. My second supervisor, *Dr Chris Mellish*, for it was his insight at a few crucial times that helped me recalibrate and finally head toward the finishing line.

I wish to thank sincerely my examiners *Dr Julie-Ann Sime* and *Dr John Levine* for their effort, kind support and insightful suggestions during the viva, and for making the viva process an exciting, delightful process.

The *EU funding body for the Training and Mobility of Researchers program* offered their financial support in the beginning of the project making the initial lift-off possible. The colleagues at GEC Marconi, *James Kwaan*, *Dr Roderick McKinnel* and especially *Dr David Scott* provided the crucial insights and invaluable technical expertise that helped enormously in my first timid steps in this work and the world of research in general.

I am also deeply grateful to the colleagues in the Approximate and Qualitative Reasoning Group, especially to *Dr Jeroen Keppens*, *Dr Finlay Smith* and *Alexios Chouchoulas* for helping kindly and unreservedly with research issues and computing technical details throughout my PhD.

My wife, *Virgínia Biris Brilhante*, is always my constant support, guidance and example. Despite being challenged to complete her PhD at the same time as I was completing mine, she offered her uncompromising love and encouragement all the way to the

end. The final result would not have been reached if it was not for her helping me face up to the circumstances no matter how challenging they seemed to be. Much more than the PhD, I owe to her who I am today.

Our daughter, *Sofia* for it was her example of growing up during the PhD years, seeing the world through the very first innocent discoveries and surprises, that reminded me that the point was not the end of the journey, but the experience of the journey itself. My *mother*, who endured not seeing me for a long time and whose prayers, love and most unselfish support keep my family and myself going.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Elias Biris*)

Table of Contents

List of Definitions

List of Figures

List of Tables

Chapter 1

Introduction

In the endeavour of understanding the complex physical systems of the world surrounding us, it is usual for scientists to represent these systems using models. Models are widely used for a variety of tasks, such as conceptualisation of the real world systems in order to facilitate reasoning about interesting aspects of these systems, prediction of a system's behaviour based on hypothetical conditions, diagnosis and design. It is also usual to build qualitative models in order to facilitate a more flexible representation of domain knowledge required for the generation of explanations.

There has been extensive work on the ways in which problem solving relates to the design and use of models. Research has shown that in human reasoning experts actually use a wide range of models according to the task at hand (Williams et al., 1983). Evidence also exists that using many different types of domain models provides better assistance in the process of learning about different concepts related to a specific topic, (Rassmussen, 1983; Sime, 1994). Using multiple models to support explanations actually involves manipulating various levels of informational detail as well as employing different strategies in order to prepare a response for a user.

This thesis explores the possibility of a modelling framework for explanation generation. In the heart of the framework lies a Compositional Modelling (CM) module (Falkenheiner and Forbus, 1991), extended to automatically generate models whilst handling uncertainty associated with user-preferences that are introduced during the explanatory interaction. In correlation to traditional CM, the framework attempts to formulate models for domain systems by assembling model fragments which represent the structure and/or behaviour parts of the targeted system. Uncertainty handling is addressed by the exploitation of a combination of mechanisms that can deal with vagueness of user preferences, and the indeterministic decision on which fragments should be selected for those parts of a domain system for which there is no evidence of preference in advance to the formulation process.

This chapter starts with an introduction into the endeavour of CM based explanation generation, with the handling of user explanation requirements in mind. This is followed by a definition of the problem for uncertainty handling in an explanation generation environment using CM. Then, a scenario for the development of this research is outlined. Lastly an outline of the thesis is given.

1.1 Compositional Modelling for Explanation Generation

CM has been traditionally regarded as a flexible and generally effective technology for the knowledge representation tasks that accompany the requirements for explanation generation. Its flexibility originates in the employment of modular, semi-independent descriptions of the pieces of knowledge that comprise the larger domain systems which are to be explained. These descriptions, also known as model fragments, can be designed so that they carry enough information to describe the phenomena that they represent, and can be selected and assembled into system models based on heuristic methods and constraints that must be satisfied prior to the fragments being admitted to the collective of fragments that comprise the overall system model.

Selecting model fragments that can, first of all, describe the phenomena of interest in an explanatory interaction, and then fit the knowledge requirements of the user requesting the explanations, is a task that CM handles well. In effect the problem of selecting those fragments that naturally fit these requirements can be formulated as a constraint satisfaction problem (Miguel and Shen, 1999), with the user requirements acting as guides in the selection of fragments. The fragments themselves can carry constraints that allow decision making related to whether a fragment's assumptions are acceptable in the company of other fragments, which may in turn have their own assumptions and information dependencies. A constraint satisfaction algorithm could be sufficient in theory.

Indeed this approach has been explored significantly in other threads of research in the CM arena, e.g. in (Nayak et al., 1992; Nayak and Joskowicz, 1996; Rickel and Porter, 1997; Keppens, 2002). However, in all these cases there still remain questions regarding the handling of the uncertainty around the preferences of the user that requests the explanations, when the interaction is viewed as part of an explanatory dialogue. In fact only (Keppens, 2002) has shown progress regarding attendance to the preferences of the user that effectively should guide the modelling process, albeit not in an explanatory interaction context. The aim of this thesis is thus to cover this gap between model formulation and tending user requirements, with targeting audiences not necessarily being model designers or CM experts who may be able to set explicit preferences to drive the constraint satisfaction algorithms.

1.2 Handling the Uncertainty of User Preferences and Modelling Selections

It is natural to consider where this much mentioned uncertainty in the user preferences comes from or what form it manifests itself in. The following are the possible sources of uncertainty:

- The determination of a user's level of understanding can only be approximate, since it is based on a prescribed set of abstract principles (e.g. attempts to classify human communication when describing objects in order to obtain abstractions of how to organise explanatory utterances in a meaningful manner for various types of indicated user understanding (Paris, 1989; Cawsey, 1993)) which, although general, cannot be deemed as fitting exactly each user's personality.

- The specification of what is "proper complexity" for the representation of the model's subparts, in order to satisfy the user's information demands and comply to the system's beliefs about the user's understanding, can only be determined in a vague manner. With the need of updating the explanation context to reflect the user's information demands and understanding, this vagueness may increase. Typically, at best it is possible to identify the most suitable representation for some of the domain system components, based on the complexity of fragments used for these components in previous interaction instances and the current explanation context. To complete the model it is necessary to be able to *propagate* the effects of selecting some fragments with an amount of certainty, throughout all of the remaining components' available fragments, and to identify suitable representations for these components.

- The mapping of all possible levels of user understanding to all possible component representations is a complex task. Due to this complexity, an exhaustive mapping is neither feasible nor desirable. On the other hand, using characteristics of the model fragments such as their resolution or precision (Leitch et al., 1999) to identify their suitability for different levels of user understanding can alleviate much of the complexity, but may well cause non-deterministic results when these mapping instances mix together.

Whilst these uncertainty sources are by no means exhaustive they are largely characteristic of the types of information vagueness that need to be handled by a program that prepares knowledge representations for the generation of explanations. In this thesis the attempt to manage these uncertainty sources leads to the exploitation of Approximate Reasoning techniques, including fuzzy logic (Pedrycz and Gomide, 1998) and Bayesian networks (Pearl, 1988). Such techniques can enable powerful yet flexible mechanisms of combining mappings of the users' understanding onto the representational complexity of the available modelling fragments, and can handle effectively the complexity of propagating the effects of indicating preference for some model fragments towards making decisions on the selection of the remaining fragments.

1.3 Research Context and Assumptions

In the discussion above, and indeed throughout this thesis, there is a clear separation of concerns for the tasks handled by the two main modules of an integrated program that generates explanations accounting for user preferences and uses CM techniques for knowledge representation, in a manner as depicted in Figure 1.1. The explanatory front end only handles requests for new explanations from the user, keeping track of the interaction and performing the information extraction from models supplied by a CM-oriented back end. The CM module, on the other hand, receives information about the selection of some of the available fragments, and attempts to complete the selection with fragments for those parts that are not yet represented, and assembles all the fragments into one model that is supplied to the front end. The Fuzzy Reasoning module, situated between the two ends mediates the information passed by the interaction with the user to the fragment selection mechanism.

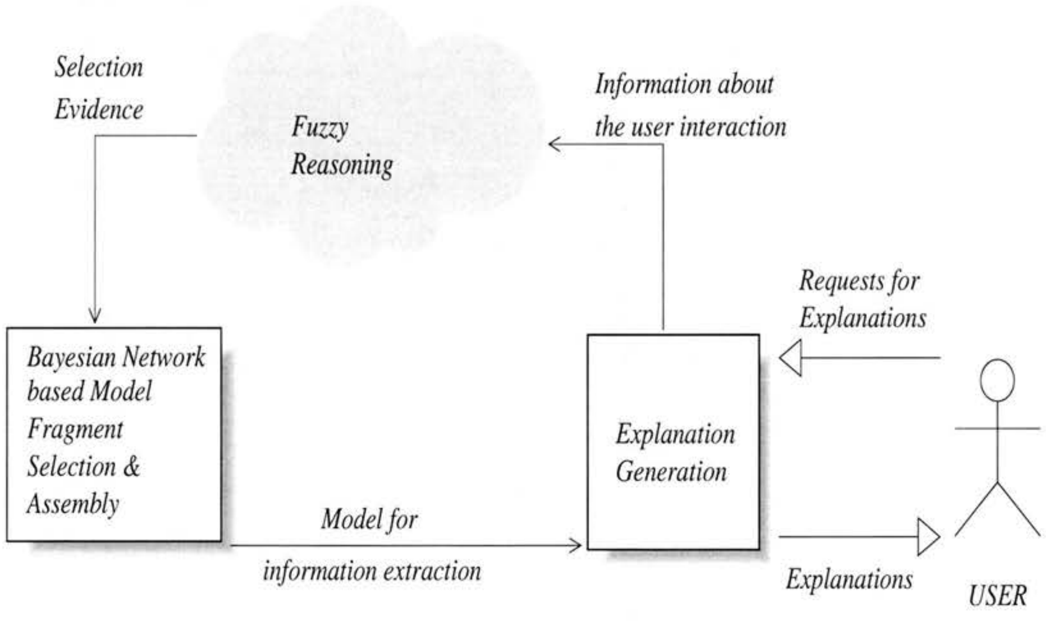

Selection Evidence
Fuzzy Reasoning
Information about the user interaction
Requests for Explanations
Bayesian Network based Model Fragment Selection & Assembly
Explanation Generation
Model for information extraction
Explanations
USER

Figure 1.1: Simplified Explanation Framework.

This depiction provides the context within which the Framework of approximate model composition for explanation generation can be investigated. This context is grounded towards modelling systems from the engineering domain. In particular, the models are qualitative models containing components of engineering systems (or processes, e.g. the Rankine cycle in engineering thermodynamics (Moran and Shapiro, 1992)).

The Framework does not attempt to introduce any new algorithm for the generation of explanations, or the propagation mechanism in the Bayesian model fragment selection or even for the inference applied by the Fuzzy Reasoning module. Instead it aims at investigating whether the combination of prominent technologies and methods from the three disciplines provides a feasible and effective mechanism to enable uncertainty handling in the explanation generation context. However, the focus of this thesis lies particularly in the Compositional Modelling work with the novel development of the model selection mechanism forming a significant part of the theoretical descriptions that follow.

1.4 Thesis Structure

The thesis consists of the following main parts:

Chapter 2: Background provides an overview of the research background in all domains involved in this research thesis: Compositional Modelling, Explanation Generation, Bayesian Networks and Fuzzy Reasoning.

Chapter 3: Framework for Model-Based Explanation Generation lays the foundations of the investigated Framework, describing the overall connectivity between its constituent modules, and exemplifying its usage.

Chapter 4: Theoretical Development of the Framework delves further into the theoretical realisation of the framework, focusing especially on the model fragment selection mechanism, exemplified through a set of test-cases. The chapter also provides a qualitative comparison of the present work with other compositional modellers.

Chapter 5: Scalability and Complexity Issues investigates the various aspects regarding the ability of the Framework to scale up for handling larger sets of model fragments or more complex model fragments, providing an empirical exploration of the expectations. The chapter concludes with a set of observations and recommendations regarding the scalability issues of the model fragment selection mechanism.

Chapter 6: Conclusion and Discussion summarises in retrospect the contributions of the thesis and provides some insight towards future extensions.

Chapter 2

Background

The focus of this chapter is to provide an overview of the research techniques developed for the tasks of automated model formulation (Section 2.1), explanation generation (Section 2.2), and the two methodologies employed for the task of Approximate Reasoning, i.e. Fuzzy Logic (Section 2.3) and Bayesian Networks (Section 2.4). Though these areas are quite broad in the number and types of research techniques that they enfold, the aim for this chapter is not to provide an exhaustive account of all the recent research trends involved in each of these areas, but to supply summaries of those particular techniques that are of importance to this work.

The last section (Section 2.5) summarises the background review, and provides an account of the main contributions of this work seen under the light of the preceding analysis of the existing research in the aforementioned areas of interest.

2.1 Model-Based Reasoning

A model is usually a representation of an aspect of the real world, known as the referent. Models are constructed for various reasons. It is usual to create simplified qualitative models in order to conceptualise the referent and facilitate reasoning about it when the aspect of the real world is too complex to handle in its entirety. Other use of such models aims to model a device while designing it, in order to experiment with its possible behaviour, thus facilitating the design process, or in an effort to provide a more flexible representation of domain knowledge required for the generation of explanations.

Studies of human reasoning indicate that experts, during problem solving, actually use a wide range of models according to the task at hand. Even for the "simple" case of reasoning about a single problem, it is common practise to use multiple models to refer to different aspects of the system under consideration, (Williams et al., 1983). In industrial processes a very influential paradigm is that of the "shop-floor", where different people like operators, supervisors and engineers with different levels of knowledge, experience and motivation cooperate to solve different problems, (Ravindranathan, 1996). The necessity of multiple models was recognised quite early by the model-based reasoning community, resulting in many systems especially for diagnosis (Gallanti et al., 1989; Struss, 1992), design (Murthy and Addanki, 1987; Sime, 1994; Ravindranathan, 1996), explanation (Falkenheiner and Forbus, 1991; Nayak, 1994; Levy et al., 1997; Liu, 1991; Nayak and Joskowicz, 1996). Especially for explanation the usage of multiple models can support multiple levels of explanation detail, scope as well as context with respect to the user's apparent needs and the history of previous discourse.

Several issues arise when dealing with multiple models. For instance, what general principles should be applied when specifying the models, or which dimensions, (Leitch et al., 1999), should be considered for the models' representation and organisation. This also relates to studying the types of relations that help to link models together in order to improve model-switching, i.e. using alternative models, during reasoning (Sime, 1994). Another significant issue is the trade-off between the effect of the assumptions made (regarding, say, when to use multiple models) and the consistency and completeness of the knowledge base, and between the effect and the efficiency of the reasoning. These and other points will be discussed in turn, starting with a set of generic modelling principles in Section 2.1.1. In Section 2.1.2 different approaches in research for model construction will be expanded while Section 2.1.3 contains further elaboration of issues regarding the usage of models.

2.1.1 Modelling Principles

A model represents (refers to) an area of the real world (referent). Typically, a model is used to infer information that cannot be revealed from the referent directly except perhaps at the expense of much greater cost, time, danger etc.

The aspects of the referent (e.g. the time-varying quantities and the dependencies between them) that the model is expected to infer are determined by the model's *purpose*. Different uses may be reflected in the relevant aspects, therefore making the purpose of the model a crucial factor in conceptualising the referent. The purpose of the model establishes also the perspective that is adopted when considering the referent. The perspective essentially restricts the perception of the referent's features: some of them are neglected and some are considered in depth. Additionally the purpose of the model determines which of the relevant aspects of the referent will be inferred from the model and which will serve as inputs for the inference procedure.

The modelling process can be seen, in general, as two-fold (Schut and Bredeweg, 1996):

- defining a representation and reasoning formalism;
- depicting a particular system in terms of such a formalism.

Also, when deciding about the representation methodology the following issues need further elaboration:

- what representation *primitives* should be required as essential;
- what model properties could serve as *dimensions* across which the different models could be represented.

Choosing the primitive concepts for the representation is essentially a definitive process for the modelling language that should be used to form models. In the case of explaining industrial processes, it is required that a model can explicitly refer to both structural and behavioural descriptions of a component in a device or system. Structural descriptions depict the hierarchical way(s) in which a component can be decomposed into sub-components, as well as the connection paths between these subcomponents. Behavioural descriptions on the other hand cover the relationships between the physical quantities related to a component. Representational primitives that could be used for these descriptions include, (Leitch et al., 1999):

- *variables* for the time dependent quantities;
- *domains* as support sets for the variables;
- *terminal variables* representing the interconnections between components;
- *parameters*, i.e. empirical (exogenous) coefficients between variables;
- *relations* between the variables used. These could take the form of equations (differential or algebraic) or in a more abstract way they could be represented as sets of if-then rules.

In addition to the choice of the representational primitives, it is necessary to define the method in which different model properties can be used as *dimensions* along which models can vary. Such dimensions serve as means of modelling taxonomy. A classification into groups of the most important and frequently used modelling dimensions is found in (Ravindranathan, 1996):

- *Ontological*: such as source of the knowledge supporting the model (empirical or theoretical), level (object-level or meta-level), orientation (implicit or explicit), scope (from a main system down to its components), resolution (amount of information contained, or, the number of observable physical parameters considered);
- *Representational*: generality (range of model applicability) and perspicuity (understandability, clarity and user-friendliness);
- *Behavioural*: i.e. related to the behaviour specification required to solve a problem. Within this category one may identify precision, i.e. a reflection of the number of distinctions supported by the description and the underlying semantics of such distinctions (e.g. quantity space). Also in this category the following can be classified: accuracy (absolute and relative), expressing the closeness of the behaviour generated to the referent's relevant behaviour, and uncertainty, which is a measure of the confidence attached to a state or behaviour.

Abstraction is an operation which may be seen as a combination of varying resolution, scope and precision, since it can be defined as altering the number of variables and relations to move from complex models to simpler ones or vice versa, and do so in a way that does not affect the causal relations within the subsystem variables, thus resulting always in "faithful" transformations, (Struss, 1992). *Aggregation* is intimately related to abstraction since it performs the grouping of the variables and relations into subsystems. By re-describing an entire system in terms of its subsystems and their interactions (shifting from part to whole) it is also possible to move across from general to specific (shifts in focus). Both abstraction and aggregation may be seen as the two dimensions of *granularity* of the modelled systems or processes, (McCalla et al., 1996).

Different definitions of dimensions are available in the relevant literature. Some of these can be regarded as supersets of the definitions give above. In most of the cases it is possible to identify the dimension(s) defined as conjunctions of dimensions previously in this section. An example of this can be found in (Davis and Hillestad, 1993): resolution is defined as a multifaceted concept covering structural properties, attributes, logical dependencies, processes, spatial and temporal properties. In this case it is relatively easy to "decompose" the term of resolution into several of the dimensions defined above: resolution for structural and attribute properties, level for logical dependencies, scope and abstraction for processes, spatial and temporal properties.

The points raised in this section seem to be generally agreed on by the majority of researchers. However, the choice of dimensions for the task of model construction and (as can be seen later) for the linking of models together, is still based on intuition rather than concrete and coherent analysis of the advantages, drawbacks and relations that are intermingled with the usage of such dimensions. Issues like the significance of independence of different dimensions are still under investigation. Also, it is very important to understand which dimensions are important for tasks like explanation generation and why. It is clear that further contributions should aim for understanding of the rationale behind the choice and definitions of such dimensions.

2.1.2 Model Construction

The problem of constructing a model could generally be stated as follows, (Schut and Bredeweg, 1996): given a particular representation and reasoning formalism in one hand and a particular system in the other, how is a model of that system constructed and how can this process be partially automated?

Model construction may be seen as depicted in Figure 2.1 (adapted from (Schut and Bredeweg, 1996)). Having defined the necessary representation and reasoning principles which should comply with the intentions, model construction proceeds, using the concepts presented in the previous section, to the creation of an instantiated model.

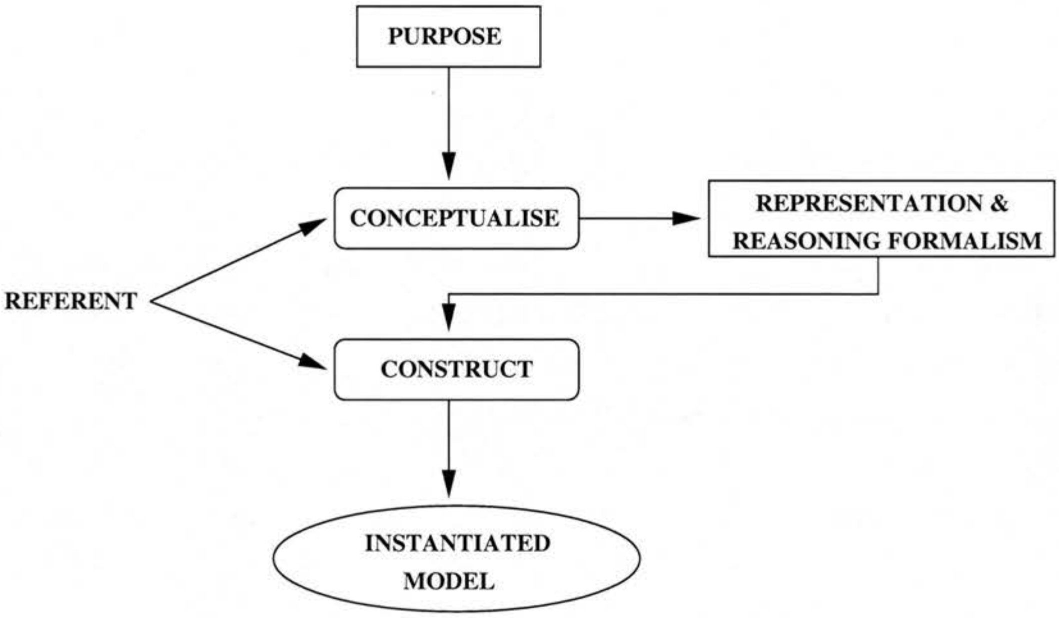


Figure 2.1: Model construction stages

The task of model construction could be divided into two subtasks:

- *model specification*: based on the aspects of the referent that are considered relevant, and with the harness imposed by the modelling principles, an initial model is specified. The specification concerns the definition of relevant variables, for each variable the set of values that it can take, and the relevant principles that govern system behaviour. Initial values are also considered, as well as whatever constraints might exist, specifying behavioural boundaries and other relative primitives such as geometrical characteristics or configuration details and properties.

- *model assessment and debugging*: having had an initial model formulated, the process continues with assessment of the behaviour prediction generated by simulation of the model. The concluded behaviour is compared to the actual behaviour of the referent. These conclusions generated by the model that correspond to the observed features of the referent are regarded to be satisfactory. Discrepancies may appear because the model can contain one or several errors, the inference engine may be designed incorrectly, or the observations and/or measurements of the system were incorrect. If such discrepancies are discovered the process continues with debugging of the model: identification of one or several errors in the model and modification of them to bring the predicted and observed behaviour in line.

The approaches followed for the task of model construction can be divided into two main categories:

- *model composition*: a model is *composed* automatically from a set of predefined partial models. This is the approach that the work of this thesis will be focused on;
- *model induction*: a model of a certain system is *inferred* from the behaviour it exhibits.

Both categories address the model construction process from specific circumstances. Nevertheless, only the former category is of interest to the research described in this thesis and will be expanded below.

2.1.2.1 Model Composition

According to this approach an explicit distinction has to be made between the knowledge that is specific to a system and the knowledge about physical principles (semi) independent of the system, (Schut and Bredeweg, 1996). The former is also referred to as *case knowledge* or *scenario* for the system under consideration, while the latter is usually known as *domain knowledge* and is represented as model fragments, i.e. the partial models. This distinction serves both modularity of knowledge representation as well as re-usability of the knowledge fragments in different scenarios.

The approaches in model composition can be divided into three major groups: *elementary* compositional modelling, *advanced* compositional modelling and *bond graph* modelling.

2.1.2.1.1 Elementary Compositional Modelling The pioneering work of *elementary compositional modelling* is reported in (DeKleer and Brown, 1984; Forbus, 1984). Although such work is not directly named as "compositional modelling", it presents the special characteristics that compositional modellers have. Model construction is similar: a structural description of the system is assumed, in the form of a device topology for ENVISION and as a scenario in Qualitative Process Theory (QPT). Additionally, a library of predefined models that capture the relevant physics of the domain is assumed to be available. In particular, the predefined models are in the form of component models in ENVISION, describing the potential behaviours that a component may demonstrate, and in the form of "individual views" (i.e. static characteristics of objects) and "processes" (dynamic aspects of objects) in QPT. A model is constructed by selecting those model fragments from the library that match the structural description of the system. It is essential for the success of such procedure, that the terms used in the device topology or scenario correspond to the terms used in the library of model fragments.

These approaches were significant since they contributed to the initial effort of defining the appropriate vocabularies for the representation of qualitative knowledge. However, they are very limited from the point of view that it is impossible to generate multiple models from one scenario. These approaches are characterised as one-to-one mappings from a scenario description to the library of model fragments. Thus, contradictory viewpoints for the same component cannot hold at the same time. The modeller that builds the model fragment library decides which components should be neglected and which should not. Eventually, phenomena can be represented with a fixed level of detail, and hence a fixed perspective and accuracy.

2.1.2.1.2 Advanced Compositional Modelling In advanced compositional modelling the aim is to overcome the drawbacks exhibited by elementary compositional modelling. The approach could be defined as a one-to-many mapping from the scenario to multiple models of the system under consideration. Different levels of detail and different perspectives are among the goals this time. The basic guideline to achieve this is by making explicit the assumptions that underly the different perspectives. Thus, representing these assumptions explicitly inside the modelling primitives is one of the essential characteristics of approaches in this category. These assumptions can also be viewed as different dimensions in modelling the referent.

The most prominent approaches that outline this area of research, and are discussed further below are:

- Compositional Modelling (CM): this is the pioneering work reported in (Falkenheiner and Forbus, 1991), which eventually lent its name to the homonymous area of automated model formulation methodologies.
- Causal Approximations, described in (Nayak, 1994; Nayak and Joskowicz, 1996), brings modifications to Forbus' CM in an attempt to attack the issues of intractability of the latter approach.
- Relevance Reasoning, presented in (Levy et al., 1997), similarly to Nayak's approach it attempts to attack intractability but follows a different methodology for formulating the initial model for a domain system.
- Dynamic Constraint Satisfaction CM, by (Keppens, 2002), which attempts to acknowledge and use preferences indicated by the user, as bias for the selection of model fragments.

In **Falkenheiner and Forbus CM**, the primary concern is to decompose domain knowledge into semi-independent model fragments, each describing various aspects of objects or physical processes and possibly bearing explicit modelling assumptions. Representation primitives are employed from QPT, Forbus (1984), though without a distinction between individual views and processes. The domain knowledge is represented as model fragments with conditional usage based on a set of assumptions (which could possibly be empty). Altering the set of assumptions different and potentially mutually contradicting perspectives on objects or processes can be represented in the same model fragment library. Collections of such mutually contradictory model fragments are defined as *assumption classes*. So, for example, it is possible to define a friction and a friction-less model of the motion of an object on a surface by using assumption classes. Importantly, assumptions are ordered in assumption classes with respect to their complexity, e.g. a friction-less model is less complex than one with the friction considered explicitly.

There are two kinds of assumptions: simplifying and operational. Simplifying assumptions make the underlying approximations, perspectives and granularity explicit, while operating assumptions state default assumptions about system behaviour for complexity management. Both kinds are used to impose only the relevant aspects of the situation for consideration, and also both apply only to individual instances of the scenario as opposed to the scenario as a whole.

The process of model construction consists of trying to find appropriate model fragments for a given scenario description and a query. The query is typically a question in English for a certain feature of the system. The process for generating the final model consists of five main steps:

Query Analysis During which the contents of the query are identified in terms of which objects, quantities and relations are included. A collection of minimal assumption environments (consistent sets of assumptions) is formed. An ATMS-based algorithm (DeKleer, 1986) is used to keep track of the sets of consistent assumptions throughout the whole process. These assumptions provide the handle to generate the set of appropriate model fragments.

Object Expansion For each of the partial environments created during query analysis, the next stage is to find which additional objects are required. The intuition is that all components relevant either explicitly to the query or implicitly to components referred in the query should be included in the resulting model.

Candidate Completion After deciding which components are relevant it is necessary to decide how to model these components. So, in this step all consistent ways of modelling the relevant components are determined. The partial environments of assumptions are completed with additional relevant modelling assumptions. An ATMS-based algorithm may be used to resolve inconsistencies.

Candidate Evaluation and Selection The set of candidate environments is pruned by retaining those with the smallest number of objects and the fewest assumptions, since these can be considered as the most basic representation environments.

Model Use and Validation The final model, containing a set of assumptions filtered by the previous steps, still cannot guarantee completeness. Analysis of the results is necessary for validity to be confirmed. The model is used for a simulation stage that may use a qualitative and/or quantitative simulation algorithm. If the outcome is either an empty prediction, or a violation of the model's assumptions then an inconsistency has been discovered.

This approach primarily targeted the provision of the required representation formulation mechanism to facilitate work in areas like handling tutorial explanations. Among its advantages are representation modularity, domain independence of the reasoning involved, simplicity and insulation from details of the domain model (which to a programmer would sound similar to data abstraction and polymorphism). However, it has some important *limitations*:

- In general, the problem of model formulation from model fragments, as it is described in (Falkenheiner and Forbus, 1991), is intractable. There are three sources of intractability: (a) deciding what phenomena to model, (b) deciding how to model the chosen phenomena, and (c) satisfying the domain independent constraints. Nayak, in (Nayak, 1994) and (Nayak and Joskowicz, 1996), attempts to solve this intractability (see below).

- Query analysis should be more elaborate so as to be able to reason with the information requirements indicated in the query, in order to determine the type of information that should be included in the model. So far there is a restriction that the needs of the modeller should in the end be expressible to a set of modelling terms that can be compared against the model fragments.

- The efficiency of the approach is limited. For tutorial sessions addressing small domain systems it might be just sufficient. However, for the provision of on-line help and continuous monitoring, the algorithm implemented following the approach would lag behind the events of the observed system.
- Finally, in the context of explanation generation, the approach does not attempt to handle the uncertainty associated with determining the desired level of representation detail based on the preferences set by the user.

Nayak's Causal Approximations work, (Nayak, 1994), provides probably the most important extension to the Forbus' Compositional Modelling. The work attempts to address CM's problem of intractability of model formulation by introducing *causal approximations* as the basis of identifying the simplest model. The definition captures the intuition that more approximate descriptions usually explain less about a phenomenon than more accurate descriptions. Hence, the causal relations entailed by a model decrease monotonically as models become simpler, leading to an efficient, polynomial time algorithm for finding adequate models.

The actual procedure generates the simplest possible model that can explain the behaviour of a given system. It requires additional information for the selection of an appropriate set of model fragments. This additional information is provided in the form of *structural* and *behavioural* constraints, and by indicating the *expected system behaviour*, (Nayak et al., 1992). Structural constraints are expressed by the physical properties and the structural relations between the components of the system that is being modelled. Behavioural constraints provide information that is not explicitly available in the structural context for a component, specifically referring to the component's behaviour and the implications it has upon the behaviour of other related components. The expected system behaviour is an abstract, possibly incomplete description of *what* the device does (but not *how* it does it), and captures what is commonly referred to as *function* of a device. Thus, given an input and an output an adequate model should be able to explain how the input affects the output. To do this the model must include a chain of modelling causal relations from input to output, connecting all intermediate quantities.

In this modified approach, model construction consists of two phases. In the first phase it searches for an adequate model. Initially, given a description of a system the algorithm extends it using quantities that appear in the expected behaviour of the system. Using the structural and behavioural contexts a set of applicable model fragments is selected, and a check is performed on whether or not the model includes a causal chain of quantities which covers the quantities from the expected behaviour. If this is true then the algorithm proceeds with the second phase. If not, the selected model fragments are adapted by selecting less general model fragments until the expected behaviour is satisfied by the emergent model. In the second phase, the adequate model is simplified by removing individual model fragments and/or replacing them by more approximate ones, until no simpler model can be found.

Further extensions are presented in (Nayak and Joskowicz, 1996) in three distinct points:

- Organisation of the model fragments in the library, by using practical class level descriptions;
- Extension of the model selection algorithm by including behaviour generation using order of magnitude reasoning (Raiman, 1991);
- Introduction of the component interaction heuristic, which allows to find an initial model that is significantly simpler than the most accurate model.

Another extension of the original Compositional Modelling approach by Falkenheiner and Forbus is the **Relevance Reasoning work of Iwasaki and Levy**, (Levy et al., 1997). The extension concerns the selection of model fragments on the basis of explicit *relevance* and *irrelevance* assumptions. The model fragments can be grouped into *composite* model fragments (or CMFs). Modelling assumptions can be attached to CMFs and may be used to distinguish alternative ways to model the same phenomenon. These assumptions include *relevance claims* connecting causally connected model fragments and those about the problem solving task, e.g. desired accuracy and temporal granularity, which determine the level of detail at which to model the relevant phenomena. For each of the quantities that appear in the query-input, the model formulation algorithm selects the simplest CMF that describes it, such that the set of the selected CMFs make consistent modelling assumptions. It differs from Nayak's approach, which chooses the most complicated model fragment for each quantity and uses a procedure to simplify the resulting model. Therefore in cases where the candidate CMFs can vary significantly in their complexity Iwasaki's algorithm can lead to significant savings in the search, since the more complicated model fragments are introduced only if necessary.

Finally, a CM approach that attempts to acknowledge and use preferences indicated by the user as bias for the selection of model fragments, is the work entitled **Dynamic Constraint Satisfaction CM with Order-of-Magnitude Preferences** presented in (Keppens, 2002). The approach has been motivated by challenging requirements of automated ecological modelling that existing techniques fail to fulfil, including the non-monotonic nature of modelling reasoning, modellers' subjective and idiosyncratic preferences for design choices among different alternatives, the diversity of model representation formalisms, and model "disaggregation" — e.g., a population partitioned into a sub-population or individuals — which does not correspond to merely adjusting the model's level of detail (or granularity) as in the physical systems domain.

The inputs to this compositional ecological modeller are a modelling scenario (e.g., describing the dynamics of ecological systems such as predator and prey populations), a library of model fragments, and some user-defined model fragment preferences. The modelling process starts with the library of fragments being exhaustively instantiated to the scenario, thus generating a space of all possible partial models. Disaggregation is used subsequently, replacing variables and/or equations by corresponding sets of more detailed variables and/or equations. At this stage, the problem of selecting and combining a set of partial models (from the model space), which is consistent and satisfies the modelling assumptions, is automatically translated into a dynamic constraint satisfaction problem (Mittal and Falkenhainer, 1990) of the activity constraint type (Miguel, 2001). These are constraint satisfaction problems (CSP) extended with special restrictions (also known as activity constraints), which are able to introduce or remove attributes and their constraints to/from the problem space. The advantage of adopting a CSP approach is that it allows the use of existing, well-studied constraint-satisfaction algorithms, avoiding the development of yet another *ad hoc* compositional modelling algorithm. User preferences for model fragments are incorporated to form a dynamic preference constraint satisfaction problem. Consequently, the model solutions yielded not only satisfy the standard modelling constraints defined but also take into account preferences of individual users.

One question that may arise from the above discussion is: "how are the model fragments constructed in the first place?" Advanced compositional modelling seems to have moved the work from defining each time a big model from scratch to defining many model fragments in a library. An important and frequent assumption is that a correct and complete model fragment library is available before-hand. However, the task of model fragment construction, in a consistent way, is still an open research issue. Further investigation is also necessary for the ability to use previously created models that might be relevant to the situation at hand, perhaps with some appropriate modifications, instead of creating a model from scratch each time a new task is being introduced. An approach that might seem useful for such a task is described in (Schut and Bredeweg, 1995), where model modification can be done semi-automatically, in interaction with the user. In addition, relatively little research has been done for relating the cost of each generated model to the considered task. The conclusion for compositional modelling approaches is that there are still some issues that need further investigation:

- How to define model fragments, i.e. how to decide what phenomena can be described in a single model fragment and what assumptions to make regarding the contents of each one. An interesting point is raised in (Schut and Bredeweg, 1996) regarding the use of methods from knowledge acquisition for model construction in cooperation with the modeller.

- How to organise model fragments in the library, and what assumptions can be made about the model fragment library.

- How to make the structure of composite models flexible and store previously created models for possible future reference.

2.1.2.1.3 Bond Graph Modelling A further approach in model composition is that of *bond - graph* modelling. This has been developed from system dynamics, (Karnopp et al., 1990). According to this approach physical systems can be conceptualised in a terms of a limited set of energy processes, thus using a very abstract level of representing a system. Consequently it can be applied to the whole range of physical systems, explaining various devices. There are, however, certain limitations. First of all the modeller must possess substantial physics knowledge, besides knowing the elementary bond graph concepts. Also, behaviour prediction becomes very difficult since a bond graph model has to be "translated" into a format suitable for simulation.

2.1.3 Use of Multiple Models

A number of important issues arise from the potential use of multiple (qualitative) models (Sime, 1994). These include:

- Should implementation involve reasoning with a large scale model or is it better to use several multiple models?
- Should the models be generated or selected from a library of pre-specified models?
- How should the selection of models be performed?
- How should the models be related (linked) together?
- How, and when to switch between models?

Each of these issues is discussed below.

2.1.3.1 One Large Scale Model or Several Models?

Representation of knowledge about a physical system may be realised in either of the following ways:

- as a large model, with different subparts focusing on different aspects of the system, or
- as pre-specified multiple models of the relevant aspects.

The Compositional Modelling approach, by Falkenheiner and Forbus, and also the relevant extensions to it (described in the previous section) favour the first representation method. In these approaches a large model is created focusing on a particular aspect of the situation or system that is being considered, instantiating only the parts that are *relevant* to the task at hand.

An alternative approach that follows the second type of knowledge representation is known as *Graphs of Models* (GoM) (Addanki et al., 1991). It is aimed at dealing with the problem of analysing physical systems, where a large amount of knowledge is required. In this work, large physical domains are decomposed into *graphs of off-the-shelf models* with nodes representing the models at different approximation levels and the edges the assumptions that must change when switching between models. Analysis can be carried out by choosing the simplest model that is an acceptable approximation of the system being looked at. The problem is how to select this appropriate model.

The algorithm that is described selects an initial model from the graph of models and generates a behaviour prediction given a description of the modelled domain system . The selected model is then assessed, by verifying if the generated predictions match with the actual measurements gathered from the real system. If there is no match then an *empirical conflict* occurs and another model has to be selected, by changing the relevant set of assumptions and therefore traverse the graph of models. Further details for the choice of models are given later on, within the discussion for model switching. The basic intuition is to search for the models that comply with certain "rules" and also to follow the edges that have a higher priority over others in order to reach the final, consistent with the problem, model.

Another approach that utilises multiple models for reasoning is presented in (Ravindranathan, 1996). The work attempts to suggest an architecture for intelligent reasoning about industrial processes, via choosing generality, precision and scope as dimensions for the different models. The models are either *procedures*, *rules* or *equations*, defined across three different levels of the generality dimension, the equations being the most general form of knowledge, rules represented as of medium generality, and procedures being the most specific (lowest generality) of all. The organisation of models is quite similar to the GoM, with the exception that models are now represented across three dimensions instead of one that is used in GoM. The choice of dimensions in the architectural design is based purely on intuition.

Both types of representation have their share in advantages and drawbacks. Building a large model poses severe limitations in size as well as computational power. For complex physical systems, manipulating a large model will undoubtedly end up defining one that has a much larger scope than what might be necessary for the current situation. On the other hand, such a large model should serve a wide variety of modelling at different levels and with varying accuracy for all the possible approximations. This makes in general the selection problem (Section 2.1.3.3) intractable, although several extensions to compositional modelling attempt to resolve the situation. Iwasaki's and Nayak's approaches, Nayak (1994); Nayak and Joskowicz (1996); Levy et al. (1997), seem to succeed in doing this through using approximations and relevance of model fragments. Nevertheless, having a larger model imposes a better control over the relation of model fragments, making it easier to transfer information between them and to combine results from their use.

GoM-like approaches have the advantage of greater flexibility, but nevertheless, need further improvements. In particular, having too many models could be the problem for this case. The initial approach was evaluated on very small scale test-beds. In more complicated structures, as the number of components increases the number of necessary assumptions increases exponentially, thus making the algorithm practically unusable for such cases. Combining the two methods in a way that not all the models need to be specified in advance, but also can be generated on the way, could provide a solution to the problem.

2.1.3.2 Generate or Select Pre-Specified Models?

Model generation is, as previously described, a task of conceptualisation of aspects of the referent (part of the real world that is being modelled). As such, it is necessary to formalise knowledge that can be used for the representation and also for inference of new knowledge about the referent, i.e. as re-usable model fragments. It is also important to decide which part(s) of the represented knowledge are necessary for the particular task. The trade-offs to be considered are the amount of assumptions that may be required by the selected representations, the impact that some of these assumptions might have on efficiency, and the completeness/consistency of the system's reasoning. Thus, the more assumptions are made about individual model fragments, the fewer constraints need to be placed on the model library as a whole and the more specific the reasoning of the system involved.

Selecting from a range of pre-specified models may be more efficient in terms of having no actual model construction task. However, it is possible that, while attempting to resolve discrepancies between the model prediction and the actual behaviour of the system, a large space of possible alternatives may have to be traversed first.

Combining both approaches could prove the most efficient solution. So, it would be possible to utilise a predefined model when it can be used for the situation at hand, or to create a different model using the relevant model fragments. The ability to modify one of the previously constructed and used models, which is quite (but not exactly) relevant for the task at hand, is also a possible improvement to the efficiency of the model selection algorithm.

Most of the approaches reviewed so far generate a single model by selecting pre-specified *model fragments* instead of selecting a pre-specified *model* (with the exception of GoM approaches). However, the ARC method (Liu, 1993) can generate not just one scenario model, but several, for a single task. In addition, the modelling assumptions for a scenario model are generated automatically during an application session and then stored in the knowledge base for possible future re-use. At any point in time, if certain assumptions become invalidated due to change in task or qualitative state of the circuit, ARC formulates a new scenario model or selects one, from the stored models, which satisfies the state. Although promising, this approach may lack efficiency since it cannot actually modify a model that has been already stored for later use. Thus, it always resolves to generating from the beginning a different model once there is no relevant (even if there is a *quite similar*) model in the model-base.

2.1.3.3 Model Selection

Either generating models from model fragments or selecting from a range of pre-specified models for the entire system to be modelled, the problem of determining selection criteria needs careful consideration. Although researchers have different standards about setting their criteria for selecting among different model fragments, the majority agree that choosing for the right model is a task-dependent process, due to the following factors:

- the inputs to the reasoning algorithm, i.e. the query or the user's response in an interaction with the system;
- the outputs, i.e. the behaviours of interest;
- the context, i.e. the structure of the device model for which individual component models are selected, as well as the behavioural context for each of the component models. For explanation, the context is also biased by the user's goals and the explainer's (system) intentions. Therefore, factors like the user's level of expertise, the contents of previous discourse play a significant role in defining a relevant context.

Selection of model fragments based on context has been applied by Compositional Modelling and its extensions. In all such cases the structural description of the system is known in advance and, together with the domain theory, it is used to find those fragments that can answer coherently and consistently a given query. Structural abstraction in (Falkenheiner and Forbus, 1991), or else causal approximations in (Nayak and Joskowicz, 1996) and relevance assumptions in (Levy et al., 1997), are used in order to select the simplest model. User requirements for information, which vary depending on the user's knowledge, can also help in the selection process by providing the requirements as to the simplicity level of the model that can be generated for answering user queries.

Drawing motivation from trying to employ multiple model reasoning for diagnosis the following model selection principle has been commonly adopted (Struss, 1992):

> "use models as simple as possible and as sophisticated as necessary".

Through a formal theory of models, Struss presents methods for *logically* relating models together by moving along different levels of model abstraction, approximation (i.e. representation) and simplification. The result provides a foundation for control strategies that start off with the simplest and cheapest of models and retracting modelling assumptions. More detailed models may then be evoked by information obtained from system observation. A set of possible faults can be detected at a time, and it is known as the *focus of suspicion* (FOS). The problem that is faced is to determine the most likely candidate to model the fault detected.

Intuition is also very important in defining the selection criteria. For example Ravindranathan in (Ravindranathan, 1996) uses a couple of principles for selecting models that can resolve a conflict/discrepancy in the reasoning of the "MuRaLi" system: increase in scope with increase in generality, and reduction in precision with increase in generality. MuRaLi uses observed parameter values, together with the interpreted events/commands to select a procedural model. If the model fails in use, the system chooses a model from the layer of rules (decreased generality) that has higher scope and lower precision than the procedural model, and through forward reasoning attempts to find a consequence that can be achieved by another procedure. If reasoning with the rules model also fails to tackle the situation, an equation model with even less precision level for its parameters is used, in an attempt to deduce additional facts for another model in the rules layer to arrive at a reasoning. If this is successful a new procedure is defined at the end of the chain of reasoning which, when executed, can succeed the original goal. If not then additional knowledge is necessary and the user is consulted.

Modelling dimensions that are involved in the selection process, is also an important issue for selecting models. Models should represent only one position along each of the dimensions, in order to preserve consistency, (Liu and Farley, 1991). Since there is no generally consistent way about defining these dimensions and no agreement as to the number of dimensions which should be employed, maintaining consistency can be a very difficult task.

2.1.3.4 Model Linkage

Linking models together is another significant issue. Two major approaches exist for relating models together. Relating models through using modelling dimensions is perhaps the most popular one in recent research, (Leitch et al., 1999). The other regards using modelling assumptions. However, most existing modelling techniques use both for linking models.

Modelling dimensions practically determine a hierarchical way of relating models. For example models can be related by abstraction and aggregation, having a hierarchical connection from the whole system down to individual components. Several dimensions have been proposed and a general consensus is to move across multiple dimensions when representing models, regarding each dimension as a separate hierarchy. Many modellers use model dimensions to link models. For instance, in Falkenheiner's Compositional Modelling, structural abstraction is used to derive from a large scale system model a simpler (i.e., smaller grain-sized) model relevant to the query.

(Iwasaki and Simon, 1994) define a wider view of abstraction, as occurring along many dimensions: structural, functional, temporal and qualitative. The issue of deriving a more abstract model from a complex dynamic model is discussed in their work. The situation is being looked upon in two ways:

- *bottom-up*, from a detailed description of the complex system to a description at a higher level of abstraction; and
- *top-down*, when modelling a complex system, a certain level of components is fixed: above that level, structure and interrelations of components are explicit;

below, the components are black boxes with no detailed internal structure.

A very important question is whether, when determining the causal structure of the model and subsequently deciding to elaborate some part of the model in greater detail, it is necessary to re-examine the causal ordering from the beginning or not. The answer drawn in (Iwasaki and Simon, 1994) is that there is no need for re-examining the causal ordering of an aggregated structure after elaborating a part of the structure. This is because the causal ordering is not sensitive to the grain-size of analysis.

Weld relates models along the "approximation" dimension in terms of accuracy of models, via an algorithm for crafting models based on a query directed shifting through model accuracy (Weld, 1990, 1992). Model accuracy is also used for linking in GoM, (Addanki et al., 1991). There, models in nodes are of different accuracy, and shifts through levels of accuracy take place by introducing and retracting approximations. These shifts can be visualised as moving upwards (towards the simplest model) and downwards (towards a maximally complex model), respectively. The discrepancies between observations of the real-world system's behaviour and the predictions of the model are used to select which approximations to retract.

Hobbs defines a notion of the granularity dimension in terms of both abstraction and aggregation (Hobbs, 1985). This leads to defining a method for constructing simple models from more complex ones, relating them though abstraction and aggregation. The idea is formalised and employed for a tutoring diagnostic system in (McCalla et al., 1996).

2.1.3.5 Model Switching

One of the important points of reasoning with multiple models is to be able to decide on how and when to switch between models. It is generally agreed that no matter how sophisticated the knowledge representation and organisation is, if there is no efficient model switching algorithm then the whole idea becomes too time consuming to implement, and even more so for complex situations.

Almost all approaches try to solve the problem of *how* to switch between models, especially the initial ones. However, little support has been provided as to *when* to switch models, in order to determine a consistent and formalised way for the task.

Ravindranathan decides to switch to a different model once there is a conflict that the model used so far cannot resolve due to lack of knowledge (Ravindranathan, 1996). The algorithm employed in MuRaLi uses these intuitive heuristics for the selection of models that can help resolve the problem at hand. However, drawbacks exist with this approach, with respect to the task of providing explanations. In particular, models are loosely linked together. There is no consistent and formalised way to define changes among them when moving from model to model or from one layer of models to another, since the whole setting up of the model space is done by intuition alone. This may give rise to a consistency question regarding the preservation of assumptions when moving between models.

Addanki on the other hand, describes an algorithm for traversing the GoM structure in a much more consistent way, (Addanki et al., 1991). Modelling assumptions are used as a basis of structuring models and are grouped into mutually exclusive classes. Model switching occurs when there is a conflict between the prediction from the model and the actual system behaviour. The importance of making the modelling assumptions explicit is also noted by Falkenheiner and Forbus, (Falkenheiner and Forbus, 1991) and Struss, (Struss, 1992). The problem is that all such approaches are too domain dependent, and therefore the generality of the criteria for model switching is limited only to the relevant tasks.

Murthy, (Murthy, 1987), describes an algorithm to switch between models with quantity spaces that vary across four levels of resolution. The heuristic used for the switching is rather simple: starting with the smallest resolution model, switch to a higher resolution model when the results of an operation are ambiguous. A significant limitation of this approach is that it is based only on one dimension and model switching happens only for efficiency, while it might be necessary to switch between models also for generating explanations at different levels of detail, or scope.

Choosing between models that span across different approximation or accuracy levels has been introduced firstly by (Weld, 1990, 1992). Model switching takes place upon the detection of model behaviour discrepancies. The model refinement process starts with the initial model that presented the discrepancy and, by eliminating one of its assumptions, generates a set of possible models. The algorithm selects the model with the first acceptable match, although it may be better if the best match would be selected.

The validation of the selected final model can be done either by carrying out internal consistency checks or by direct observation to compare the model predictions against the actually measured behaviours. Both approaches, however, are flawed. The first is due to the impossibility to perform *complete* consistency checks, and also because such checks are domain dependent, posing an extra burden on a human modeller for providing a comprehensive set of checking rules. The direct observation approach has the problem that it does not indicate which model will actually alleviate the problem. So, an important weakness of the whole algorithm is its inability to indicate which model to switch to; it can only suggest that the current model is inappropriate.

Weld suggests a possible algorithm that can address this problem (Weld, 1992) based on a generate and test methodology. When a discrepancy is discovered, a generator enumerates the models that eliminate at least one assumption held by the current model, and the tester determines whether a candidate model can account for the discrepancy. If the candidate model resolves the problem, then analysis continues with the new model. The solution though is far from perfect, since for large models the search space of candidate models can become exponentially large (as GoM is used for the representation of models this drawback is attributed to it). The cost of model switching needs also to be addressed.

A different approach is followed by Sime, in an attempt to employ multiple modelling for reasoning in tutoring systems (Sime, 1994). The actual switching of models is based on *Cognitive Flexibility Theory* (Spiro et al., 1988). Switching between models is, therefore, focused on tutoring principles such as whether or not adjacent tutorial cases should overlap. The potential of the approach is significant for explanation since there is a clear intention to try to "teach" the user about a concept.

2.2 Explanation Generation

The term "explanation" signifies different things to different people. An explanation could be generally deemed to be an elaboration of a concept to facilitate its understanding. In scientific research for example an explanation could be the analysis of the rationale behind an observation, whereas in everyday life it could be a mere conveyance of knowledge so as to make something clear to someone who seeks more information about it. The term may even be used to refer to an interactive information exchange where both participants should come to a shared understanding or as a knowledge transaction with one predominant knowledge conveyor.

Within the knowledge-based systems research the notion of explanation is similarly diverse. Early work on MYCIN (Buchanan and Shortliffe, 1984) introduced a traditional approach to explanation as providing answers to "how" and "why" questions, by tracing the reasoning of the system. "How" questions were interpreted as "how did you conclude that" whereas "why" questions as "why did you ask me that". Based on a full retracing of the reasoning path MYCIN could provide answers to such questions.

This early work emphasised the significance of providing explanations that would support the rationale behind the reasoning of an intelligent system. Further work either focused on this "traditional" approach, in an effort to improve the type of responses given back to the user (as in (Wick et al., 1995)), or faced the problem of explanation generation more or less disjointly from the line of reasoning of the corresponding knowledge based system, considering better ways to communicate complex information more naturally and effectively (Mittal and Paris, 1995; Forsythe, 1995).

In its simplest form the explanation problem may be stated as follows. Given a communicative goal, find information from the expert system's knowledge sources that is relevant to achieving this goal and organise this knowledge into a coherent multi-sentential text (Moore, 1995). Additionally, in order to allow the explainer to be responsive to the user's feedback and sensitive to previous explanations, the system should be able to reason about its own previous responses. Such capabilities should be facilitated by a representation of what the system was trying to communicate and how it was supposed to do that. Furthermore, Cawsey (Cawsey, 1993) and Moore (Moore, 1995) emphasise the necessity of modelling the user to tailor the explanations according to the user profile, and also to the handling of follow-up questions and the detection of possible misconceptions.

Based on the above general statement for the explanation generation task, a methodology for explanation generation can be identified to have the following distinct points:

- There is a concept not understood or unclear (i.e. identification of the needs for explanation);
- There is a coherent piece of knowledge to convey (explanation content),
- The process is intrinsically interactive and the generated explanatory text should comply with the general principles which govern the generation of multi-sentential text. This is seen here differently than the generation of explanatory manuals. Whilst the latter task involves larger pieces of text, the explanations sought after in this work are shorter, addressing immediate information needs regarding only sub-parts (components) of the domain system of interest, for which the user asks questions. Furthermore, this interactivity introduces the need for varying the detail of the underlying representations used for explanation content extraction to correspond the perceived user understanding during the interaction. Such progressive variation of representation detail is supported naturally in automated modelling approaches, such as Compositional Modelling seen in Section 2.1.2.1.
- The process terminates with the satisfactory level of understanding from the user (this requires building, maintaining and updating a user model), a point closely related to the organisation of the content for the explanations;
- Explanations and the problem solving process of the corresponding knowledge-based system are closely related (this demands the determination of the types of this relation and how they affect the structure and effect of generated explanations).

The following discussions will further elaborate these points.

2.2.1 Determining Explanation Needs

Every effective explanation system should satisfy its target users' needs for explanation. However, despite the obviousness of this statement, little work has been done regarding the discovery of a user's needs for knowledge in order to provide necessary and relevant information. The majority of the explanation generation systems that have been developed so far base their reasoning on analysis of human "expert" explanations or on the intuition of the designers and/or researchers who develop them. For example, Cawsey (Cawsey, 1993) describes the construction of an interactive explanation system, by analysing transcripts from the corpus of human expert-novice explanatory dialogues of circuit behaviour.

Both intuition and analysis of human-human interactions have played an important role in creating computer-based explanatory dialogues. However, they may be inadequate to fully identify the explanation needs of a particular application and class of users. In supporting explanation construction, based on analyses of human-human interactions, it is probable that the human expert might not accurately cover the user's needs. Instead, he/she will at best be able to generate an approximation of the expected explanation structure and content, probably after an incremental cycle of analysing the explanatory needs elicited from a number of human subjects, adjusting the automatically generated explanatory content and structure to meet those needs, and reevaluating the generated explanations with the target users. Cases that justify this claim can be found especially in the field of medical applications (Forsythe, 1995), where the information provided by the expert rarely satisfies fully the needs of the users (usually patients). For industrial applications, the theoretical knowledge of an expert, say, in process control may not cover exactly the practical needs of the operator of a console that monitors the various processes, in an emergency situation. Additionally, the requirements and possibilities of computer-based systems are likely to be different to those of a human explainer. Explanation systems that are based on knowledge engineer's own intuition, on the other hand, may very well end up being designs conforming too much with the designer's needs for explanation than with those of a possible user. Despite not being able to accurately determine the explanation needs of all types of users, the incremental process of assessing the user explanation needs followed by adjusting and evaluating the structure and content of the automatically generated explanations could be employed to expand the capabilities of an explanation generation program so that it approximatively meets the needs of new user groups.

There is no simple way to assess in advance what the needs are for explanation given particular applications and classes of users. As Cawsey suggests: "a mixture of techniques may be required, including analysis of human explanations, interviews with potential users and evaluation of prototypes" (Cawsey, 1995). In (Forsythe, 1995) and (De Rossis et al., 1995) a detailed analysis of human explanations supported by interviews with potential users is used in order to determine the users' needs. Forsythe specifically proposes the exploitation of ethnography, a method in anthropological research for gathering descriptive qualitative information about complex real-world settings, as an alternative for obtaining reliable data on the needs and characteristics of future users. This should be applied before making any decision about the actual design of an explanation generation system. Engaging principles of this method in the design of an explanation system leads to a re-evaluation of designers' initial assumptions and may even contribute to a radical rethinking of the system design. The success of such techniques suggests that the development of practical explanation programs should be based on both intuition, and the incremental process of defining the explanation needs of the prospective class of users, adjusting the explanation generation program to meet these needs, and validating the produced explanations against the target users.

2.2.2 Explanation Content

Having decided about the general definitive dimensions for the explanation needs of the prospective users, the next task is to determine the content of individual explanations given particular queries or based on apparent needs. The main points that regulate the explanation content are related to the explanation context. Although the details of what is meant by "context" differ among researchers, the commonly accepted points are given as follows:

- The reasoning of the problem solving process.

- A record of the knowledge that has been shared in prior dialogue (a discourse history model built up by recording the discourse salient objects and segments).

- The system's "beliefs" about the user's domain knowledge. A model of the user's level of expertise is constructed incrementally as the interaction proceeds. This model can be used to adjust the subsequent explanation content (e.g. different information should be provided to an expert user than to a novice).

- The knowledge that will be expressed in other segments of the same explanation (plans for explanations to come).

- The organisation of the domain knowledge in the system (explainer).

Mittal and Paris (Mittal and Paris, 1995) analyse the first three points and describe how these aspects of context can affect the explanations provided. These authors attempt to develop planning operators which create the structure of different parts of the explanations' text, in a way that allows reference to the explanation context. The aspects of context involve the problem solving situation, the participants engaged in the process, the mode of interaction for the communication (e.g. tutorial-like or reaching a shared understanding), the discourse history and finally, the characteristics of the external world (structure of the underlying knowledge). In summary, the explanation text depends on which planning operators are invoked and this in turn depends on the relevant context. A potential problem of this approach, however, is complicating the structure of the planning rules which use these operators, when trying to capture all aspects of the given context.

The separation of the different aspects of context may be inherently difficult, due to unclear identification of the roles of the participants (e.g. if none is clearly the explainer). To improve this a multi-stage hybrid architecture, as presented by Suthers in (Suthers, 1991), provides a possible solution. Suthers considers the task of explanation as consisting of associative, exploitive, inferential, opportunistic, prescriptive and restrictive subtasks modelled as distinct mechanisms and, therefore, provides a way to separate and catch different sources and types of context. Nevertheless, this approach sacrifices simplicity for the sake of diversity and efficiency with the increase of text-planning when dealing with complex situations.

2.2.3 Methods for Generating Explanations

Deciding about the method to generate explanations is based on the decision of what to include in them and how to structure the content of the generated text. Cawsey (Cawsey, 1993) identifies the following principles governing the structure and content of generated multi-sentential text:

- Cohesion, or in other words surface-level ties between elements in the text (also pointed by Halliday and Hasan (Halliday and Hasan, 1989));
- Coherence, the consistent conceptual relations hold between concepts or ideas expressed in different parts of a text;
- Intentionality, or the goal of the explainer must be achieved by the generated text;
- Acceptability, which relates to the hearer's (user) attitude to the generated text according to his/her own goals;
- Informativity, corresponding to the hearer's knowledge;
- Situationality, making a text relevant to its situation of occurrence;
- Conventionality, dealing with conventions that relate with texts of the specific type;
- Extra-linguistic Devices: using diagrammatic elements or other additional material in order to change the content and structure of the text generated.

These principles, together with the representation of the user requirements for explanations and the discourse history of a user's previous explanatory interactions, can help define the structure and content of the explanation text. Flexibility should be considered. A text generation module for an explanation provision component should be able to make different choices about the structure and content of the text based on one or more of the discourse situation aspects.

A simple way to do this was addressed by the first-generation explanation systems. Satisfactory explanations could be produced with techniques for traversing, pruning and translating the system's execution trace (Buchanan and Shortliffe, 1984), and by the use of hand-crafted pieces of textual information ("canned" text). Improvements on them focused on the quality of the explanations, based on better representation of the domain and explanation knowledge as well as on modularisation of the design by separating domain and explanation knowledge, (Clancey and Letsinger, 1981; Swartout, 1983; Swartout et al., 1991).

Further improvements regarding the quality and naturalness of utterances can be achieved by exploiting the linguistic side of the explanation generation. Instead of simply translating code into English sentences, an explanation facility must employ sophisticated text-planning techniques.

Some early work on text planning regarded the recognition of common patterns of organisation in texts of certain types (McKeown, 1985). Descriptions of objects, for example, follow generally several patterns such as giving details of an object's constituents. In the very simplest of forms these patterns could be represented as templates to be filled with specific items. For example, a general object-description template may receive object attributes every time it is used. This, however, imposes the restriction of using the same templates for perhaps wide ranges of cases. To avoid this restriction more flexible patterns, e.g. *schemata* (Paris, 1989), may be employed.

McKeown, for example, used a wide range of object descriptions to derive four schemata, capturing the different structures observed in the relevant textual descriptions (such as the *constituency* schema used to formalise descriptions about an object's constituents). In so doing, these schemata can help decide about a methodology to handle the user's needs for explanations and thus generate coherent multi-sentential texts (McKeown, 1985).

Paris on the other hand, focused on how descriptions of physical devices depended on the class of user they were aimed at. By comparing descriptions aimed to be given to novices with those aimed at experts, abstracting schemata can be discovered, thereby capturing the structure of the texts in each category. In particular, device descriptions aimed at novices should be based on schemata that describe the process of the device, giving some attribute information about the parts being introduced in the process. In the case of expert users, however, as such users can infer process from structure, given general knowledge from the domain they should be provided with descriptions based on schemata which reveal the constituents of the devices used (Paris, 1989).

The schemata technique provides with a fairly practical and effective way of generating coherent texts based on classification of patterns in a wide range of similar texts. However, it suffers from the same problems that canned-text approaches do. Instead of having to provide large hand-crafted pieces of text, classes of patterns have to be provided. The number of such classes can be rather large when dealing with complex cases. Thus, a serious overhead to the whole process may appear. Additionally schemata once defined are rigid, reducing the system's flexibility.

To avoid this problem it is necessary to consider seriously why texts have different content and structure in different situations (Cawsey, 1993). Unfortunately there seems to be no comprehensive theory describing exactly how these are related. If, however, there is a way to reason about how people change their level of understanding about something, or their goals about understanding, it may be possible to generate a sequence of appropriate utterances, given some communicative goal and knowledge about the user's current mental state.

In Cawsey's system, there is no concept of "planning operators" for the text generation. An incremental "hierarchical" planning technique is applied for generating explanations about the behaviour of simple electrical circuits. Two sets of planning rules exist: those that determine the overall content of the dialogue, and those that determine the structure of the dialogue. The structure of the dialogue is based on results from previous work on the structure of tutorial discourses by Sinclair and Coulthard (Sinclair and Coulthard, 1975). Following definitions in their model, Cawsey organises the rules about the structure of the dialogue through:

- *transactions*, i.e. high level dialogue structures on some topic;
- *exchanges*, forming the transactions, which stand for general (coarse-grain) linguistic actions like opening and closing subtopic discussions, and carry out information conveyance on the subtopic;
- *moves* within the exchanges (such as an initiating move from the system at the beginning of an explanation followed by a response move from the user and a feedback move from the system);
- *acts*, providing the necessary elements for meaningful sentences, including markers ("ok" or "well"), starters (statements directing attention to a topic), metacomments (statements referring to future discourse: "what I am going to do is to go through ..."), etc.

An explanation plan is developed from the top level explanation goal down to simple graphical or textual actions, using the rules to expand goals into subgoals. As soon as primitive actions are planned they are immediately realised, while an agenda of partially ordered explanation goals is used to represent the remaining explanation yet to be generated (what is planned to be explained in the future). The explanation context influences the planning process by affecting the decision of which goal to retrieve from the agenda, which planning rule to select, and which subgoals to attempt to satisfy. The context includes assumptions about the user's knowledge, the current object in focus and the roles of the participants. All this information helps tailor explanations to the specific users needs as well as to choose between different alternatives on aspects of the dialogue.

This type of planning may not be as powerful as other traditional AI planning techniques (Reiter et al., 1995), but the trade-off is in efficiency, where the algorithm is fairly fast and also appears consistent (in most cases) with the approach which human experts follow.

2.3 Fuzzy Reasoning

This work employs Fuzzy Reasoning as means of mapping the perceived user understanding and preferences for explanation detail to the required representation detail for domain objects that the user is interested in getting explanations for. As such the work involved does not attempt to break new ground in the theory of Fuzzy Logic, but only applies the existing groundwork of the principles defined by Fuzzy Sets and Fuzzy Reasoning theories in the context of the framework developed for this research. Due to this, the following discussion of Fuzzy Reasoning principles will be only an outline as opposed to extensive in depth coverage of the work involved in that area.

Fuzzy reasoning, as realised through fuzzy set theory (Pedrycz and Gomide, 1998), provides an efficient way to deal with *vague*, *imprecise* or *ill-defined* knowledge in a human-like fashion, within a well defined mathematical formalism. In traditional approaches for reasoning without human interaction, it is necessary to handle the system's inputs and outputs and to further define concepts related to these, like the transfer function, frequency response, state-space models, in an exact numerical way. However, when the interaction with the human factor exists, inputs and outputs cannot be handled efficiently in a numerical way. This is because the human stimulations (inputs) and replies (outputs) are better expressed and understood in a linguistic way (words).

For the present research, a fuzzy logic approximate reasoner is proposed to be used as bridge between the explanation generation and the model-based simulation processes, in order to control their interaction (more details in Section 3.2.2). Among the intentions is to enable this reasoning module to select the best model that can relate to the context reflecting the user's needs, the system's intentions, and the context of the explanatory interaction so far. Choosing a model is not something straightforward as there is a certain time restriction to provide the answer (which calls for efficiency). Difficulty also arises because of having to choose models based on *all* of the above relevant context details, and to perform simplifications in different levels at each stage of understanding of the user. Fuzzy logic control may help by enabling a more "human" way of dealing with knowledge of this kind for the user, the process and the object of interest, making better suggestions for explanation, than conventional crisp rule-based controllers.

For the sake of completeness, a brief outline of the theory is given below. A full treatment of it can be found in (Zadeh, 1965).

2.3.1 Basic Theory

Fuzzy logic, as introduced by Zadeh (Zadeh, 1965), is based on the refutation of the so called *principle of dichotomy*. Having a set of objects A, an object x either belongs to A or not. This can be stated in a more formal way by using the concept of *membership function* of x in a subset A of U, the universe of discourse:

$$\mu_A(x) = \left\{ \begin{array}{lcl} 1 & : & x \in A \\ 0 & : & x \notin A \end{array} \right.$$

The difference in fuzzy logic is that instead of having $\mu : A \rightarrow \{0,1\}$ we have $\mu : A \rightarrow [0,1]$.

Based on this changed notion of membership value, it is possible to define a mapping of the type:

$$\mu_A(x) : A \rightarrow [0,1]$$

by which x is assigned a number in $[0,1]$, indicating the extent to which x belongs to (has the property) A. Using this concept information for imprecise and incomplete knowledge can be represented in an exact mathematical way.

Operations on fuzzy sets can also be defined to allow combinations of sets, i.e. combinations of concepts. These operations are respectively:

$$\begin{array}{rcl} \text{Complement } \bar{A} & : & \mu_{\bar{A}}(x) = 1 - \mu_A(x) \\ \text{Union (OR) } A \cup B & : & \mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \\ \text{Intersection (AND) } A \cap B & : & \mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \end{array}$$

Fuzzy rules are based on the above concepts of fuzzy sets and their operations. An example of a linguistic fuzzy rule is

IF (input is high) THEN (output should be low)

Using fuzzy sets for linguistic values, such as "high" and "low" in the above example, rules like this can be represented as a 2-dimensional fuzzy set R in the product space $A \times B$, where A and B are the domains for "input" and "output" (for simplicity x and y here):

$$\mu_R(x,y) = \mu_{A \times B}(x,y) = \min_{x,y}\{\mu_A(x), \mu_B(y)\}$$

providing the degree to which R is true.

In the case of having a set of N rules, a set of R_i can be obtained from each of them, with $i = 1, 2, \ldots, N$. Combining all the results can be performed by union that produces an overall set:

$$R = \bigcup_{i=1}^{N} R_i$$

A very important principle in fuzzy logic is the *extension principle*, that helps in "translating" classical mathematical formulae and laws into their "fuzzy" equivalent types. It can be defined as follows:

Definition 2.3.1 (Extension Principle) *Let $X_1, X_2, \ldots, X_k$ be universes of discourse, $X = X_1 \times X_2 \times \ldots \times X_k$ be their Cartesian product, $A_1, A_2, \ldots, A_k$ be fuzzy subsets respectively defined in X_i, $i = 1, 2, \ldots, k$, and $f : X_1 \times X_2 \times \ldots \times X_k \rightarrow Y$ with $y = f(x_1, \ldots, x_k)$ be a crisp (i.e. not fuzzy) function. Then the* extension principle *transfers the fuzziness of $A_1, A_2, \ldots, A_k$ into a fuzzy set B of Y where*

$$B = \{(y, \mu_B(y)) \mid y = f(x_1, \ldots, x_k) \in X_1 \times X_2 \times \ldots \times X_k\}$$

and

$$\mu_Y(y) = \begin{cases} \sup\limits_{(x_1,\ldots,x_k) \in f^{-1}(y)} \min\{\mu_{A_1}(x), \ldots, \mu_{A_k}(x) & , \quad f^{-1}(y) \neq \emptyset \\ 0 & , \quad f^{-1}(y) = \emptyset \end{cases}$$

where $\emptyset$ denotes the empty set.

A very important concept in fuzzy logic which takes a significant step into overcoming the difficulty in modelling complex problems is that of a *linguistic variable*. A linguistic variable is one which has values that are words or sentences or propositions in a natural or artificial language. Its (formal) definition connects closely to the following principles, (Lee, 1990; Driankov et al., 1996):

$$\langle x, \mathcal{L}x, \mathcal{X}, \mathcal{M}\chi \rangle$$

In this quintuple, x denotes the variable's name, such as *age, temperature, height, speed, error,* etc. $\mathcal{L}x$ is the set of linguistic values (or *quantity space*) that x can take. For example, in the case that x is *age* A it may have the following linguistic value set:

$$\mathcal{L}A = \{young,\ middle\_aged,\ old\ \}.$$

$\mathcal{X}$ is the actual physical domain, or the *universe of discourse*. For example in the case of a linguistic variable like *annual_income* this can be the range $[£10000, £20000]$. The *universe of discourse* can be either discrete or continuous, depending on the variable that is under consideration.

Finally, $\mathcal{M}\chi$ is a semantic function giving the meaning of a linguistic value in terms of the quantitative elements of $\mathcal{X}$. This function takes as input a symbol, e.g. *old* and returns the meaning in terms of a fuzzy set (Driankov et al., 1996).

Finally, all of the above concepts can be used for development of fuzzy state models, giving an estimate for the output, based on inadequate information for the input and also on information about the relation between the input and the output. The tool to achieve this is *fuzzy composition*. Given a fuzzy relation R from U to V, and a fuzzy subset A of U, a fuzzy subset V is inferred given the max-min compositional rule:

$$\mu_B(y) = \max_x \{\min\{\mu_R(x,y), \mu_A(x)\}\}$$

where $U = \{x\}$ and $V = \{y\}$; this is usually denoted as $B = A \circ R$.

2.4 Bayesian Networks

Using networks to represent knowledge has been employed quite rigorously for graphical visualisation of the interdependencies between related pieces of knowledge. Much work has been done in developing formal syntaxes and semantics for such representation schemes, including associative networks and conceptual graphs. Of particular interest to the present work is the network representation which depicts the degrees of belief on propositions and their causal dependencies upon other propositions, and which supports the inferencing by propagating evidence throughout such networks, termed Bayesian networks (Pearl, 1988).

2.4.1 Main Concepts

A Bayesian network is first and foremost a directed acyclic graph (DAG) in which the nodes represent the variables of interest, and the links stand for the causal dependencies between variables (Pearl, 1988; Verma and Pearl, 1990; Pearl, 1993). The links are labelled with conditional probabilities that provide estimations of the strength of the dependencies. Furthermore, a Bayesian network is a compact, localised representation of a probabilistic model. The key to its locality is that, given a graphical structure representing the dependencies (and, implicitly, conditional independences) among a set of variables, the joint probability distribution over the set can be completely described by specifying the appropriate set of marginal and conditional distributions over the variables involved (Pearl, 1988). Equally important, the graphical structure can be used to guide processing when evaluating queries against the model.

Formally, for a vector of variables $\overrightarrow{X} = \{X_1, \ldots, X_n\}$, respectively taking their values from finite sets $\mathbf{X}_1, \ldots, \mathbf{X}_n$, a Bayesian network can be defined as a DAG with nodes representing the variables $\overrightarrow{X}$, and a set of conditional probability distributions $\{P_i(X_i \mid \{X_j\}_{j \in PA_{X_i}})\}_{i=1}^{n}$, also referred to as the "parameters" of the network, labelling individual links between each X_i and its "parent" nodes, denoted as PA_{X_i}.

These parent nodes are X_i's direct predecessors, with an arc joining each of them to X_i.

Furthermore, they render X_i independent of all its other predecessors (in other words, knowing the values of other preceding variables becomes redundant once the values of the variables that belong in the set of PA_{X_i} are known). These nodes are also known as X_i's *Markovian* parents (Pearl, 1988, 1996).

The joint distribution can be derived as a factor of all the network's parameters (Pearl, 1988):

$$P(\overrightarrow{X}) = \prod_{i=1}^{n} P_i(X_i \mid \{X_j\}_{j \in PA_{X_i}}).$$

Bayesian networks have the following advantageous characteristic of the knowledge representation when compared to classical probabilistic representations. The joint probability distribution is calculated by considering *only* conditional probabilities with respect to "parent" nodes for each of the nodes examined. In contrast, in the classical probabilistic approach *all possible combinations* of variable assignments are considered in order to compute the joint distribution. Thus, Bayesian networks can compactly represent distributions over large state spaces.

Using the joint distribution, all probabilistic queries (e.g. finding the most likely value of a variable based on evidence of the values of other variables) can be answered consistently and coherently within the rationale of Bayesian probabilistic calculus.

2.4.1.1 An Illustrative Example

For visualisation, in Figure 2.2 a Bayesian network is given to represent beliefs in the states of a simple electronic circuit consisting of AND, OR and NOT gates. The network describes the causal relationship between the inputs (sources) and outputs of all components: S_1 and S_2 stand for the inputs to the system, while N_1, N_2, A_1, A_2, L are the outputs of the NOT, AND and OR gates. Note that the Bayesian network depicted functions only as an estimator of which variable assignments are responsible for whatever evidence provided.

Nodes that are linked in the Bayesian network signify causal dependencies between the corresponding variables. Missing network links stand for the lack of influence

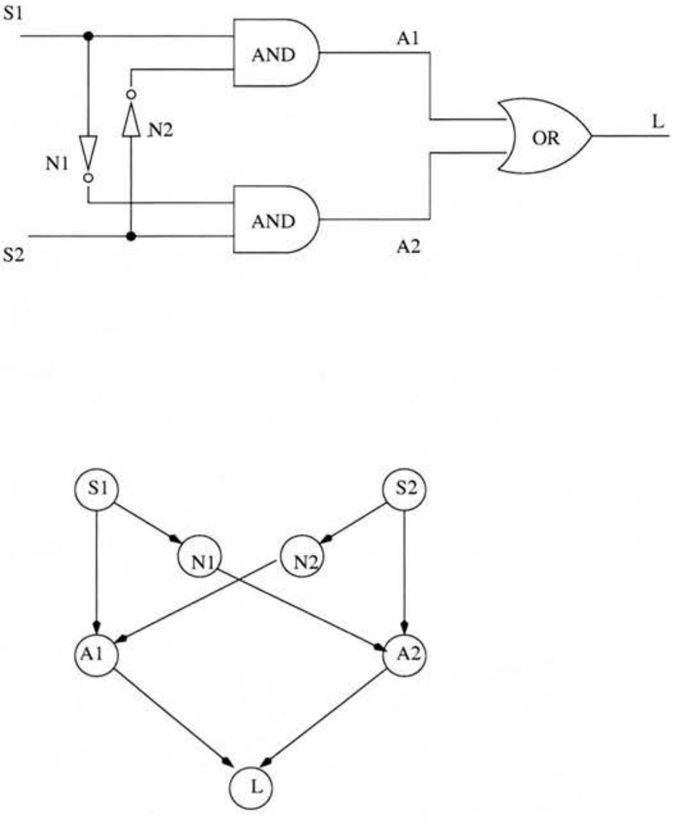


Figure 2.2: A simple circuit and its representation with a Bayesian Network

between the corresponding terminals in the system model. The links of the network are labelled by conditional probabilities and can be defined based on the correlations between the terminals, e.g. between a gate's inputs to define its output. For example, the conditional probabilities associated with an AND A$_i$ ($i = 1,\ 2$), given its inputs S$_i$ and N$_j$ ($j = 1,\ 2and\ j \neq i$) and its output are defined as:

$$P\left(A_i \mid N_j,\, S_i\right) = \begin{cases} 1 & : \text{ when } A_i = N_j \wedge S_i \\ 0 & : \text{ otherwise} \end{cases}$$

2.4.1.2 Major Advantages

Employing graphical means for representation and manipulation of probabilistic knowledge, in Bayesian networks, helps overcoming many of the shortcomings (conceptual and computational) that accompany most traditional rule-based systems. The main advantages stemming from this integration of graphical and probabilistic information are, (Pearl, 1988):

1. Graphical methods facilitate the building and maintenance of probabilistic knowledge bases that are consistent and complete. They also enable modularity in procedures for knowledge acquisition reducing the number of assessments required.

2. Graphical representations provide ways to improve computational efficiency, by supporting distributed computations (examples are provided in Section 2.4.3). By extending the basic methodologies to deal with feedback loops, essentially most network topologies can be used for inference purposes.

3. Due to the expressiveness of graphical representations, conditional probabilistic independences can be created, read, assessed or reasoned with, much easier in a Bayesian network. Probabilistic information on the other hand provides with clear preferred semantics that allows such networks to be processed coherently to accomplish given application tasks, including diagnosis, learning, explanation and others that require inference under uncertainty. Moreover, causal information can be visually checked, analysed and reasoned with, making both prediction and abduction possible within one network.

Having introduced briefly the basic properties attributed to Bayesian networks the following sections give an account of the most important methodologies for structuring Bayesian networks, and the main techniques developed for inference with such networks. Available software for reasoning with and learning Bayesian networks, is not discussed herein. An extensive list of software for general inference on probabilistic networks is maintained on the World Wide Web (Almond, 1995) and the Association for Uncertainty in Artificial Intelligence (http://www.auai.org/).

2.4.2 Construction of Bayesian Networks

Developing a Bayesian network usually requires the following decisions:

- choosing nodes/variables for the network representing the problem under consideration;

- establishing links between nodes, with respect to available (qualitative or quantitative) information for the causal interrelations and/or probabilistic conditional independence relations between the underlying variables;
- adhering probabilistic measures for the links and the root nodes, i.e. choosing the network's parameters.

The entire process of developing a Bayesian network can either be performed manually, or can be automated through algorithms designed for learning the structure and parameters of the network from available data. These issues are reviewed below.

2.4.2.1 Choosing Network Nodes

A key issue is to establish what the nodes of the network under construction stand for, and how each node should be related to others. This may be addressed by splitting the problem at hand into separate subproblems (Mahoney and Laskey, 1996). Under this view, a Bayesian network can be seen as consisting of several modules, the *local distributions*. Each local distribution consists of a variable, its parent nodes, and a set of conditional distributions for the node, given the possible combinations of values for its parents.

Decomposition into subproblems can become a very entangled task for high complexity problem cases, which may involve several hundreds (or possibly thousands) of variables. In such cases, a single level of decomposition becomes insufficient, and it may be of crucial importance to identify intermediate decompositions into sets of coupled subnetworks, each representing a partially separable component of the problem. The general principles to perform such decompositions are for these components to be both *semantically separable* and *formally separable* (Mahoney and Laskey, 1996). The former implies that the subproblems into which the problem is decomposed are *meaningful* to the expert and also posed at a natural level of detail, while the latter entails the capability of the subproblems to be re-aggregated into a complete and consistent probability model.

2.4.2.2 Establishing Links between Nodes

In principle, given the set of chosen variables $\overrightarrow{X} = \{X_1, \ldots, X_n\}$, ordered with a fixed ordering d, and a joint distribution $P(X_1, \ldots, X_n)$, a Bayesian network can be constructed recursively, to represent the distribution using the conditional independence relationships as a guide for establishing links between the nodes (Pearl, 1996).

For instance, starting with the pair (X_1, X_2), an arrow from X_1 to X_2 is drawn if the two variables are dependent (the arrow pointing to X_2 if X_2 depends on X_1, or vice versa). Continuing to X_3, no arrow is drawn if it is independent of $\{X_1, X_2\}$; otherwise examine whether X_3 renders X_2 irrelevant, once X_1 is determined, or the other way around, whether X_1 is rendered irrelevant if X_2 is known. Note that the terms "independence" and "irrelevance" are herein used interchangeably. In the first case an arrow from X_1 to X_3 is created, while in the second the arrow is created to point from X_2 to X_3. If both X_1 and X_2 are relevant, with respect to X_3, then both are linked to node X_3. Generally, at the i-th stage, node X_i is selected and directed links are drawn with direction toward X_i from its corresponding parent nodes PA_{X_i}.

At each step in this process conditional probabilities are assigned to each of the links from node X_i to its (Markovian) parents. The conditional probabilities are defined in accordance with the joint distribution $P(X_1, \ldots, X_n)$. For those nodes X_j that have no predecessors, i.e. the network's root nodes, marginal probabilities $P(X_j)$ are assigned, again as dictated by the joint distribution. The result of this process is a directed acyclic graph that models the distribution $P(X_1, \ldots, X_n)$.

The resulting graph can be characterised as either "singly-connected" or "multiply-connected", depending whether there is only one or many possible paths between any two of the graph nodes. A path here means a sequence of consecutive nodes (of any directionality) in the graph being constructed. A trade-off lies herein: although multiply connected networks allow the modelling of the underlying distribution more accurately they have computational as well as conceptual disadvantages. Exact inference procedures (Section 2.4.3) are in the worst case computationally intractable over multiply connected networks (Cooper, 1990). Moreover, even if an approximation algorithm is used for the structure of a complex network (e.g. stochastic simulation methods as described later), highly connected networks still require the storage and estimation of a large number of conditional probability parameters. Also, they can be rather difficult to understand conceptually. Therefore, although more accuracy in inferencing may be achieved by multiply connected networks, their computational practicality can be largely degraded.

The joint probability distribution of the whole network can be related to the conditional probabilities that are used to label the network's links. Given for every node X_i the probability distribution of the group of links between X_i and its parent nodes PA_{X_i}, $P(X_i \mid PA_{X_i})$, then it can be proven (Pearl, 1988) that:

$$P(X_i \mid PA_{X_i}) = P(X_i \mid X_1, \ldots, X_{i-1}) \quad , \quad PA_{X_i} \subseteq \{X_1, X_2, \ldots, X_{i-1}\}$$

This is the definition of treating a Bayesian network as a carrier of conditional independence information along the specific ordering d of the network variables. From this formula, the joint probability distribution can be decomposed using the classical probabilistic chain rule formula, to the product:

$$\begin{aligned} P(X_1, \ldots, X_n) &= P(X_n \mid X_{n-1}, \ldots, X_1) P(X_{n-1} \mid X_{n-2}, \ldots, X_1) \\ &\quad \cdots P(X_3 \mid X_2, X_1) P(X_2 \mid X_1) P(X_1) \\ &= \prod_i P(X_i \mid PA_{X_i}) \end{aligned} \tag{2.1}$$

Conversely, for every distribution that can be written as the above product, an ordering d can be identified which will help produce a DAG G to reflect the distribution (Pearl, 1988). When a probability distribution P can be decomposed into a product of conditional probabilities that reflect the conditional independence relationships as depicted in a DAG G, then G and P are said to be *compatible*.

With respect to the example of Figure 2.2, for instance, it can be derived that the joint probability distribution for the network, which is compatible with the DAG that represents the Bayesian network for the system at hand, is as follows:

$$\begin{aligned} P(S_1, S_2, N_1, N_2, A_1, A_2, L) &= P(L \mid A_1, A_2) P(A_1 \mid S_1, N_2) \\ &\quad P(A_2 \mid S_2, N_1) P(N_1 \mid S_1) P(N_2 \mid S_2) \\ &\quad P(S_2) P(S_1) \end{aligned} \tag{2.2}$$

The above product indicates the probabilistic independence relations encoded within the network representation.

2.4.2.3 D-separation Criterion

The identification of independences that need to be satisfied, in order for a probability distribution P to be compatible with a DAG G, is facilitated by the *d-separation* criterion (Pearl, 1988), described below.

In essence this criterion dictates that in order to test whether X is probabilistically independent of Y given Z in the distribution represented by a DAG G, it is necessary to verify if the node corresponding to variable Z *blocks* all paths from node X to node Y. Briefly, the criterion states that:

Definition 2.4.1 (D-Separation Criterion) *A path p is said to be* d-separated *(or blocked) by a set of nodes Z iff (Pearl, 1988):*

1. *p contains a chain* $i \longrightarrow j \longrightarrow k$ *or a fork* $i \longleftarrow j \longrightarrow k$ *such that the middle node j is in Z, or*

2. *p contains an inverted fork* $i \longrightarrow j \longleftarrow k$ *such that neither the middle node j nor any of its descendants (in G) are in Z.*

If X, Y and Z are three disjoint subsets of nodes in a DAG G, then Z is said to d-separate *X from Y, denoted as* $(X \perp\!\!\!\perp Y \mid Z)_G$*, iff Z d-separates* every *path from a node in X to a node in Y (Pearl, 1988; Pearl and Dechter, 1996).*

The d-separation criterion has been proved to be both sound and complete with respect to the set of distributions that can be represented by a DAG G (Geiger and Pearl, 1988). This entails a one-to-one relationship between the independences which are implied in the decomposed form of the joint probability distribution in Eqn. 2.1 and the node subsets X, Z, Y as determined from the DAG and the d-separation criterion. The notion of d-separation is formally related to that of probabilistic independence, denoted as $(X \perp Y \mid Z)_P$, through the following definition:

Definition 2.4.2 (D-Separation and Probabilistic Independence) *For any three disjoint subsets of nodes $\{X,Y,Z\}$ in a DAG G, and for all probability functions P the following holds (Verma and Pearl, 1988):*

1. $(X \perp\!\!\!\perp Y \mid Z)_G \Longrightarrow (X \perp Y \mid Z)_P$ *whenever P and G are compatible, and*

2. *if* $(X \perp Y \mid Z)_P$ *holds in all distributions compatible with G, then* $(X \perp\!\!\!\perp Y \mid Z)_G$

As an example of the utility of the d-separation criterion, in the DAG of Figure 2.2 the set $Z = \{S_1, N_1\}$ d-separates $X = \{A_1\}$ from $Y = \{A_2\}$. The path $A_1 \longleftarrow S_1 \longrightarrow N_1 \longrightarrow A_2$ is *blocked* by $\{S_1, N_1\}$, while the path $A_1 \longrightarrow L \longleftarrow A_2$ is also blocked since L is not in Z. However, the set $Z = \{S_1, N_1, L\}$ does not d-separate $X = \{A_1\}$ from $Y = \{A_2\}$ since L is in Z, i.e. the path $A_1 \longrightarrow L \longleftarrow A_2$ is rendered *active* by virtue of L. These two statements allow the terms $P(A_1 \mid S_1), P(A_2 \mid N_2), P(L \mid A_1, A_2)$ to be included in the joint probability distribution expression (as given in Eqn. 2.2). They also indicate that terms like $P(A_1 \mid A_2)$ or $P(A_1 \mid L), P(A_2 \mid L)$ should not be included since such dependencies do not hold with respect to the graph.

Having identified how the graphical information about the connectivity of the network nodes reflects which conditional independence relations are significant to determine the joint probability distribution, it is time to see how probabilistic distributions are elicited in the first place.

2.4.2.4 Eliciting Probabilistic Distributions

Realistically, probabilistic distributions can be difficult to obtain. It is generally recognised that it is hard to determine the parameters *which can be encoded* in a Bayesian network. Usually, available knowledge may be classified into one of the following categories (Bhatnagar, 1994):

- a number of databases, each consisting of cases recorded in various contexts of the regarded domain; and

- qualitative or quantitative probabilistic knowledge about the distribution and/or the dependencies entailed between the variables of interest.

From statistical data (if available), information on the actual mathematical form of the distribution and the inter-variable dependencies may be extracted (Lee, 1997). From such information conditional probability functions can be formulated under the specific contexts that dominate the corresponding data. The joint probability distribution, then, reflects the context under which the corresponding database was formed.

The knowledge sources in the second of the two categories listed above can serve as alternative means to acquire knowledge for a given problem, when data is not available or it is corrupted (incomplete, highly imprecise), or as a complement to existing knowledge from data. Obtainable information, however, from these knowledge sources may not be directly amenable to encoding in a Bayesian network (Druzdzel and van der Gaag, 1995). For instance, existing knowledge may not be numerical in nature: an expert may be certain of the fact that some values of a statistical variable A make some values of a variable B more likely to occur, or may have an idea of the upper and lower bounds on the numerical strength of this influence, yet may not be able to give exact numbers. Thus, information available may range from numerical point- and interval-valued probabilities, through order of magnitude estimates, to purely qualitative statements regarding the independence between variables.

When only knowledge of the second category is available, the network can still be constructed following the procedure described in the beginning of Section 2.4.2.2, (Pearl, 1988). Of course, the network developer must identify the probabilistic information of the network's links and root nodes (Pearl, 1988). To specify consistently the probabilistic information of the network links, certain functions $F_i(X_i, PA_{X_i})$ between each variable and its parents are required to hold such that:

$$\sum_{x_i} F_i(x_i, PA_{X_i}) \quad = \quad 1,$$

where $0 \leq F_i(x_i, PA_{X_i}) \leq 1$ and the summation ranges over the domain of each considered variable X_i. The conditional probabilities of the network links are then assigned as $P(x_i \mid PA_{X_i}) = F_i(x_i, PA_{X_i})$ (Pearl, 1988).

Imprecision will certainly creep in these estimates of the network's probabilistic information, largely due to false approximations or simplifications of the functions F_i, or perhaps due to omitting some of the root nodes. In (Pradhan et al., 1996; Pearl, 1988), however, it has been argued and proven experimentally (for networks used in diagnostic tasks) that imprecision in the estimation of the probabilistic information of a network can be tolerated if the *correct* conditional independence relations have been established within the network's structure. In other words, if the structure is correct then the network's performance will not be impaired significantly during diagnosis. Nevertheless, the more precise the estimation for the F_i is the more correct the inference results using the Bayesian network will be.

Various schemes have been developed in order to handle various types of knowledge used to identify the F_i functions. Typically, most of these strategies can handle the encoding of only a few (if more than one) types: quantitative (Pearl, 1988; Whittaker, 1990; Lee, 1997), partial numerical specifications (allowing for interval rather than point probabilities) (Breese and Fertig, 1991; Coletti, 1994), order of magnitude estimates (Goldszmidt and Pearl, 1992), or purely qualitative knowledge (Wellman, 1990).

A unifying method that allows manipulation of all available types of knowledge has been attempted in (Druzdzel and van der Gaag, 1995). The principal idea is that the set of all possible joint distributions for a network can be considered as points in a *hyper-space*. The true, yet unknown, distribution Pr over the network variables is then a point in that space. Within this representation, available information (qualitative and quantitative) about Pr can be represented as constraints on the hyper-space. These constraints can be translated into an abstract format, thus providing a consolidating representation scheme for both quantitative and qualitative knowledge for the domain. With each point a *second-order* probability distribution can be derived, representing the total probability of the underlying first-order distribution being the true distribution. All second-order distributions can be elicited by selecting randomly points from the hyper-space and ranking them according to their compatibility to the hyper-space constraints. At the end of this process the first-order distribution with the highest rank can be selected, or the most probable points can be treated as a new start for further refinements.

2.4.3 Inference with Bayesian Networks

Due to their diversity in supporting top-down as well as bottom-up reasoning mechanisms, Bayesian networks can be used for many tasks involving evidential reasoning, such as answering simple or composite queries regarding the truth of certain propositions, answering queries of the type "what-if" with extra constraints on the values of variables, or revising belief commitments that collectively provide a best explanation of the available evidence. The information typically required to accomplish these tasks is the joint probability distribution, encoded in a Bayesian network by the conditional probabilities of the network's links and the marginal probabilities of the root nodes. The methods devised to accomplish these tasks can be categorised into three classes (Pearl, 1988; Poole, 1996):

1. Exact methods that exploit the structure of the network to allow efficient propagation of evidence, e.g. (Pearl, 1988; Lauritzen and Spiegelhalter, 1988).

2. Stochastic simulation methods that give estimates of probabilities by generating samples of instantiations of the network, e.g. (Henrion, 1986; Hrycej, 1990; Kanazawa et al., 1995; Hulme, 1995).

3. Search-based approximation techniques that traverse through a space of possible values to estimate probabilities, e.g. (Henrion, 1991; D' Ambrosio, 1992; Poole, 1996).

The distinguishing characteristic between these is the way that reasoning is performed with respect to what is exploited the most: the network's structure or the distributions encoded in the network. Thus in the first class more significance is given to exploitation of the structural characteristics of the network, the third class moves attention to using the encoded distributions, and the second uses both with preference moved to the structural details. A brief account of the main characteristics of the first of these classes of inference techniques will be provided in the subsequent sections. Only the first class is considered as it is the class of inference mechanisms where the propagation algorithm for the model fragment selection, which is described later in this thesis, belongs to.

2.4.3.1 Fusion and Propagation of Evidence

Inferencing within this class of reasoning methods is performed by "fusion" and "propagation" of each new piece of evidence through the entire network. The goal is to eventually have an assignment of probability measures for each proposition in the network, consistent with the axioms of probability theory. Thus, each time when something new is found about one of the network variables (i.e. new evidence about an event happening or not, a reading for the value of a physical variable, etc.) the perturbation caused by this new evidence is propagated via local message passing between neighbouring nodes in the network, with minimal external supervision. Once the network reaches an equilibrium, all nodes have a consistent assignment of belief measure. A query regarding the probability of specific variables taking specific values can then be answered. Such queries could be related to requests for interpreting the effects of the input data, or requests to recommend the best courses of action (Pearl, 1988).

The algorithms which perform exact evaluation of the incoming evidence, propagating messages throughout the network to update the nodes' local belief measures include the Kim-Pearl *poly-tree algorithm* (Pearl, 1988) and Spiegelhalter's *clique-tree algorithm* (Lauritzen and Spiegelhalter, 1988).

(Pearl, 1986, 1988) describe a local message passing scheme for inferring the states of the nodes in *singly-connected* Bayesian networks. The goal for this scheme is to calculate the probability that the corresponding node is in a particular state, e.g. for a variable X_i with values v_{ik}, $i = 1, \dots, n$ and $k = 1, \dots, r_i$ $(r_i \geq 2)$ the goal is to find the probability $P(X_i = v_{ij} \mid E)$ with respect to evidence E. In the literature, this is referred to as calculating the *belief function* for node X_i.

Updating the belief functions in the face of the evidence can be accomplished by naive enumeration, (Pearl, 1988). However, this is a solution which takes exponential computational time and storage space (in the number of the network nodes that are uninstantiated). Furthermore, it does not exploit the decomposable nature of the network's joint probability distribution. It is this decomposition that suggests that any algorithm related to the inference about individual node beliefs should lend itself to parallel dis-

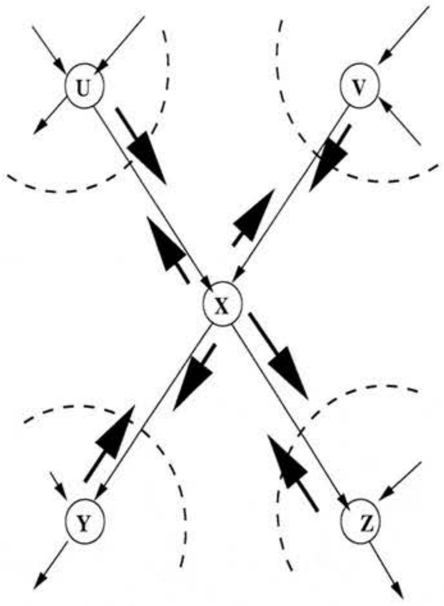


Figure 2.3: Bayesian network inference: propagation of messages in poly-trees.

tributed processing at the nodes of the network.

Informally, the message passing algorithm for *belief update* works with every node sending a “message” to its neighbouring nodes. The message actually stands for a probability vector. Suppose that a node X has two parents U,V and two child nodes Y,Z (Figure 2.3). Roughly speaking, when X is activated to update its parameters, it simultaneously inspects the incoming messages from its parents and children, computes its own belief measure by combining the probabilities carried by these messages to its own probability distribution and normalising the result, and communicates new messages along each of the links back to its parents and children. The message that X sends to Y is, essentially, the probability that Y is in a particular state, given the information that X knows but Y does not. This message is a combination of the messages that X receives from all its neighbours *except* Y, multiplied by the conditional probability matrix corresponding to the set of links from X to its parent nodes. Similarly, Y takes this message and, after it computes its own belief measure, it prepares messages for its neighbours. This procedure is repeated for all nodes in the network, in parallel.

More formally, assume that the probability distribution modelled by the network, for an ordering d of the network variables, is factorised into the form:

$$P(\overrightarrow{X}) = \prod_i P(X_i \mid PA_{X_i}).$$

For each node X_i and its immediate parents PA_{X_i} a matrix M can be attached which gives the conditional probabilities for each value of X_i within each combination of values of its parent variables. Given this matrix the *belief update* process can be described as follows:

- When X_i is activated by incoming messages, combine all these messages, by multiplying the message vectors element by element. The result is a vector v.
- Multiply v with matrix M. Normalise the result so that the product $M \cdot v$ sums to 1. The normalised vector is the *belief measure* for the variable X_i.
- Bottom-up propagation: X_i prepares new messages to send to its parents. For each parent the message is prepared by multiplying the product of the messages received from all the other parents of X_i, i.e. all parents *except* from the one for which the message is being prepared. The result is then multiplied with matrix M, and the message is sent to the designated parent node. The exclusion of the parent that is the receiver of the message from the aforementioned multiplication is done so that there is no double-counting of information for that particular parent (otherwise this parent node would receive back the same message that it sent to X_i combined with the messages from the other parents).
- Top-down propagation: X_i communicates new messages to X_i's children. Each of these messages is generated by dividing the belief measure for X_i over the message received from X_i's child which is about to receive a new message.

The procedure is initialised with all message vectors set to $(1, 1, \ldots, 1)$. Evidence nodes do not receive messages and they always transmit the same message vector i.e. $(0, \ldots, 0, 1, 0, \ldots, 0)$ with the 1 standing for the certainty (truth) of the actual observed value for the evidence node. Propagation ends when either a root node with a single child or a leaf node with a single parent is reached. Single port nodes act as absorption barriers: updating messages received through these ports get absorbed and do not cause subsequent outgoing messages.

Belief update will give the probability of every node being in a particular state given all the evidence. If, however, each node is set to the state that maximises its own belief measure, the global sequence obtained will not necessarily maximise the posterior probability of the nodes attaining the corresponding values (Pearl, 1988). In order to obtain the most likely state assignments for all nodes, a revised belief function needs to be calculated. This is done within the *belief revision* procedure. This procedure is identical to the belief update method described above, with only difference being the matrix multiplications in the steps of the belief update sequence. Specifically, the sum operator is replaced with the maximum operator. The result is a sequence of assignments for the variables, providing best explanation for the evidence. If there are more than one such assignment combination yielding the same support to the evidence, then a random tie breaking mechanism is required.

To enforce a selection of variable assignments that lead to the same optimal explanation of the evidence, local pointers should be saved with each node to mark the neighbours' values at which the maximisation is achieved (Pearl, 1988). This scheme can also be useful for retrieving *second-best* explanations and also to facilitate *sensitivity analysis* performed when it is necessary to determine the merit of testing an unknown variable.

This message passing algorithm is not so straightforward when the network is not singly connected, i.e. when there are more than one possible paths between any two of th network's nodes. Techniques have been developed however, to manipulate networks that are not singly connected to become singly connected, in order to perform the propagation of messages safely. These methods essentially may involve one of the following:

- *clustering*, i.e. forming compound variables, so that a network is transformed into a singly connected one;

- *conditioning*, which effectively helps breaking the loops by sequentially instantiating a selected group of variables to all possible values, propagating the effects to the rest of the network, and collecting the results to provide complete descriptions;

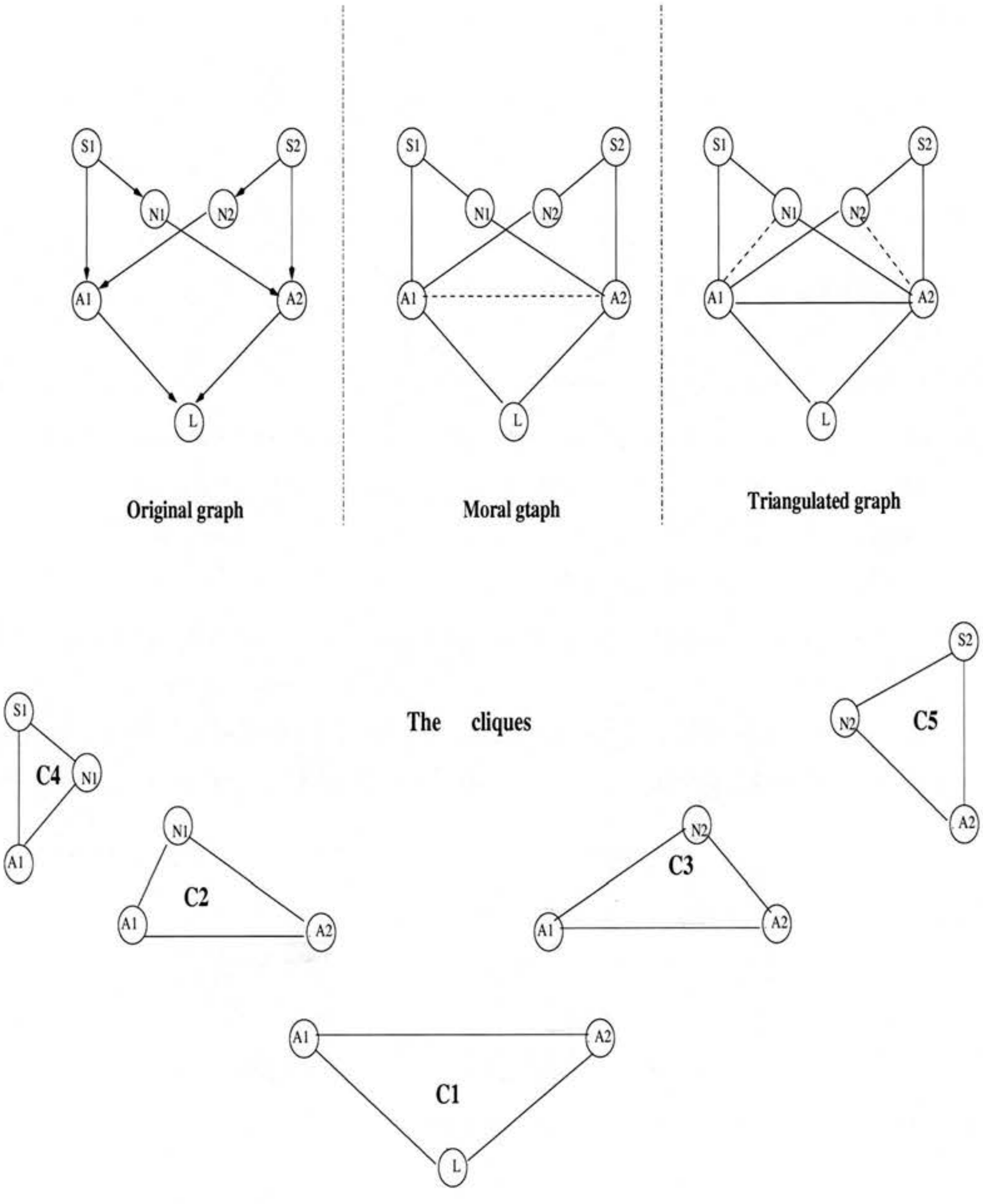


Figure 2.4: The generation of a triangulated graph, and the final cliques

- *stochastic simulation*, assigning each variable a definite value and making each node to inspect the current state of its neighbours, computing the belief distribution of its host variable and then selecting a value at random from the computed distribution. Then, beliefs are computed by recording the percentage of times that each node selects a given value.

Spiegelhalter's "clique-tree" algorithm is the second main landmark in this class of network structure based exact reasoning methodologies. It converts a network into a *tree* of clusters, or sets of nodes. Probability distributions are then calculated for the clusters during a message propagation process and, after this process, individual node probabilities can be calculated from the cluster distributions.

To formulate the cluster tree, the algorithm performs a number of transformations to the topology of the original network. First, the orientation of the arcs is dropped, and the parents with the same children are connected together with undirected links. This process, also known as *moralisation* yields an undirected, connected graph, the *moral* graph. After this step, the moral graph is transformed to a *triangulated graph* and the vertices are ranked according to a *maximum cardinality search.* A graph is triangulated if every cycle with number of nodes greater than three has a *chord*, i.e. a pair of non-consecutive nodes connected by an edge. The maximum cardinality search is an ordering procedure. Start at an arbitrary node, giving it the number 1; number the remaining nodes consecutively, and choose the node with the maximum number of previously numbered neighbours at each step. Ties should be broken arbitrarily.

As an example, the transition from original graph to the moral and triangulated versions, of the DAG in Figure 2.2 is illustrated in Figure 2.4.

Finally, cliques are determined in the obtained tree, resulting in a clique tree (Figure 2.4). A clique in a graph $G = (N,E)$ is a maximal subset of nodes such that every pair of clique nodes are connected via a graph edge. The cliques are ranked in order of their highest numbered node. Moralising the network assumes that a probability distribution for a node and its parents will belong to a single clique. Triangulation eliminates cycles of length four nodes or more. Maximum cardinality search helps to find a clique ordering with a running intersection property. The nodes of a clique that are also contained in previous cliques are all contained in *one* previous clique, the current clique's parent clique. This is formally denoted as follows (with C_i representing the cliques):

$$C_i \cap \ (C_{i-1} \cup C_{i-2} \cup \ldots \cup C_1) \ \subseteq C_j$$
$$1 \leq j \leq (i-1)$$

Let S_i denote the nodes of a clique that are also contained in a clique above (a parent clique), i.e. $S_i = C_i \cap (C_{i-1} \cup C_{i-2} \cup \ldots \cup C_1)$, and $R_i = C_i \backslash S_i$ denote the rest of the nodes in the clique ($\backslash$ is subtraction for sets). In essence the S_i are the nodes upon which the R_i nodes depend, probabilistically. Then for each clique:

$$P(C_1 \cup \ldots \cup C_i) \quad = \quad P(R_i \mid S_i) P(C_1 \cup \ldots \cup C_{i-1})$$

That is, the joint probability distribution can be decomposed to probability distributions for each clique in the transformed graph.

The propagation of messages in this transformed graph involves sending information about the probability distribution of each clique to the cliques that follow, in the ordering specified, via the clique intersections. When evidence is presented for some of the nodes within a clique, the evidence nodes are *absorbed* in the network. The probability distributions involving each observed term are projected onto either a new, reduced clique, or into a neighbour if a clique is removed. A new distribution is calculated for the new graph that results from absorbing evidence. This distribution can be used (through proper marginalisation) to derive the probabilities of the remaining nodes.

The same process of absorption/projection is applied when it is necessary to test a hypothesis for a variable taking a specific value. The hypothesised node is absorbed to the network, the clique that it was involved projects its probability distribution, conditioned on the variable acquiring the specific value, onto the next clique which, in turn, after updating its own distribution, projects it onto the next clique, and so on and so forth until the last clique in the graph. At the end the joint distribution for each of the cliques in the graph resulting from the absorption of any nodes, can be marginalised to discover the posterior probabilities of the nodes included in the clique. The main advantage of this algorithm is its ability to handle multiply connected networks without having to employ any of the techniques that the poly-tree algorithm needs to transform this kind of network to singly connected ones.

Although, these algorithms are fairly efficient for handling relatively small-sized belief networks, and they have provided excellent solutions to many real-world problems, their applicability is limited because the inference involved is NP-complete (Cooper, 1990). Furthermore, evaluation of a Bayesian network within probably approximately correct bounds is also NP-hard (Dagum and Luby, 1993).

2.4.3.2 The Shenoy-Shafer Belief-Function Propagation Method

A particular variation of the original "clique tree" inference algorithm is the valuation method inference algorithm introduced by Shenoy and Shafer, reported in detail in (Shenoy and Shafer, 1988; Shenoy, 1997). This is particularly interesting to this research since it improves greatly the efficiency of the original "clique tree" propagation, as well as introduces a greater flexibility in handling alternative types of uncertainty.

To give a brief account of its characteristics assume that the network of Figure 2.5 is given as the representation of the probabilistic dependence of variables $A1, A2, B1, C1$ on each other. The parameters indicating the prior probabilities in this network are:

- $\alpha 1$ for $\{A1\}$, to represent the prior probability distribution for $\{A1\}$.
- $\beta 11$ for $\{A1, B1\}$ represents the conditional probability distribution of $B1$ given $A1$, $\beta 12$ for $\{A2, B1\}$ represents the conditional probability distribution of $B1$ given $A2$.
- $\beta 1$ stands for the combination of $\beta 11$, $\beta 12$, i.e. the conditional distribution of $B1$ given both $A1$ and $A2$.
- $\gamma 1$ for $\{B1, C1\}$ indicates the conditional probability distribution of $C1$ given $B1$.

Suppose that evidence for $\{C1\}$ is provided, represented as probability λ_{C1} of variable $C1$ taking a particular value. The problem is to compute the posterior joint probability distribution for $A1, A2, B1, C1$, determining the effect of the evidence to the information represented in the network.

Let θ denote the set of prior estimations for the probability distributions associated with each network variable, including any given evidence. The combination, i.e. point-wise multiplication for probability theory, of all members in θ gives the joint posterior probability distribution, which is denoted as $\phi = \otimes \theta$. In the example, $\theta =$

$\{\alpha 1, \alpha 2, \beta 1, \gamma 1, \lambda_{C1}\}$ and the joint distribution is the unnormalised posterior joint distribution for $A1, A2, B1, C1$. The problem is to find the marginal of the joint valuation for subsets of interest, i.e. $\phi^{\downarrow r}$, where r is a subset of the network variables whose marginal is desired. Probability theory defines marginalisation of a joint distribution over a set of variables as the operation that reduces the domain of the distribution, by summing all possible evaluations of the distribution for all possible instantiations of the variables over which marginalisation occurs.

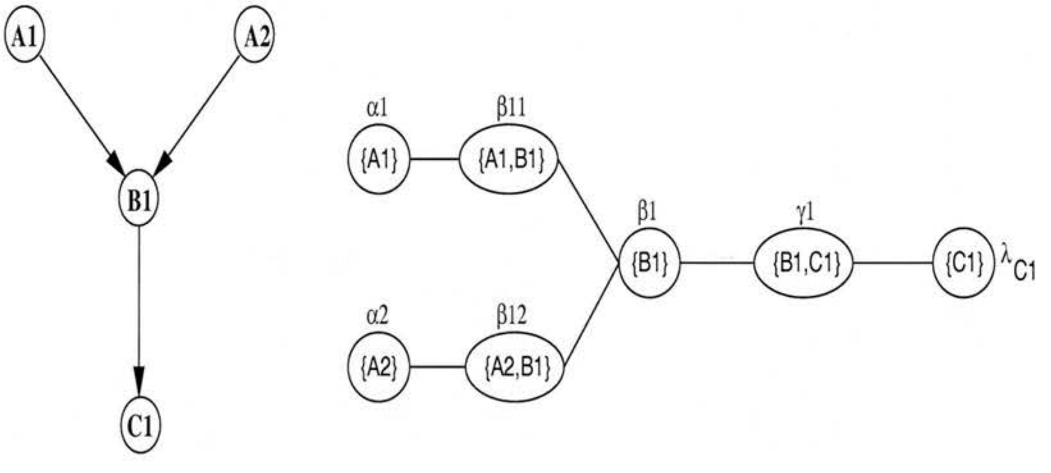


Figure 2.5: A simple Bayesian network and the corresponding join tree.

The probabilistic distributions for each of the variables involved in the initial network, as well as the associations between these variables, form a hyper-graph H (Shenoy and Shafer, 1988). For the example, $H = \{\{A1\}, \{A2\}, \{A1, A2, B1\}, \{B1\}, \{B1, C1\}, \{C1\}\}$. The first stage in applying the Shenoy-Shafer reasoning algorithm is the *join tree construction phase*. In this stage, the subsets in H are first arranged in a binary join tree (Shenoy and Shafer, 1988) and then the singleton subsets whose marginals are to be computed are added if not already included (Shenoy, 1997). After the binary join tree has been constructed each distribution in θ is associated with the corresponding node in the tree, as shown in Figure 2.5.

The second stage is known as the *message-passing phase*, when each node sends some information to each of its neighbour nodes. Each message that a node sends carries

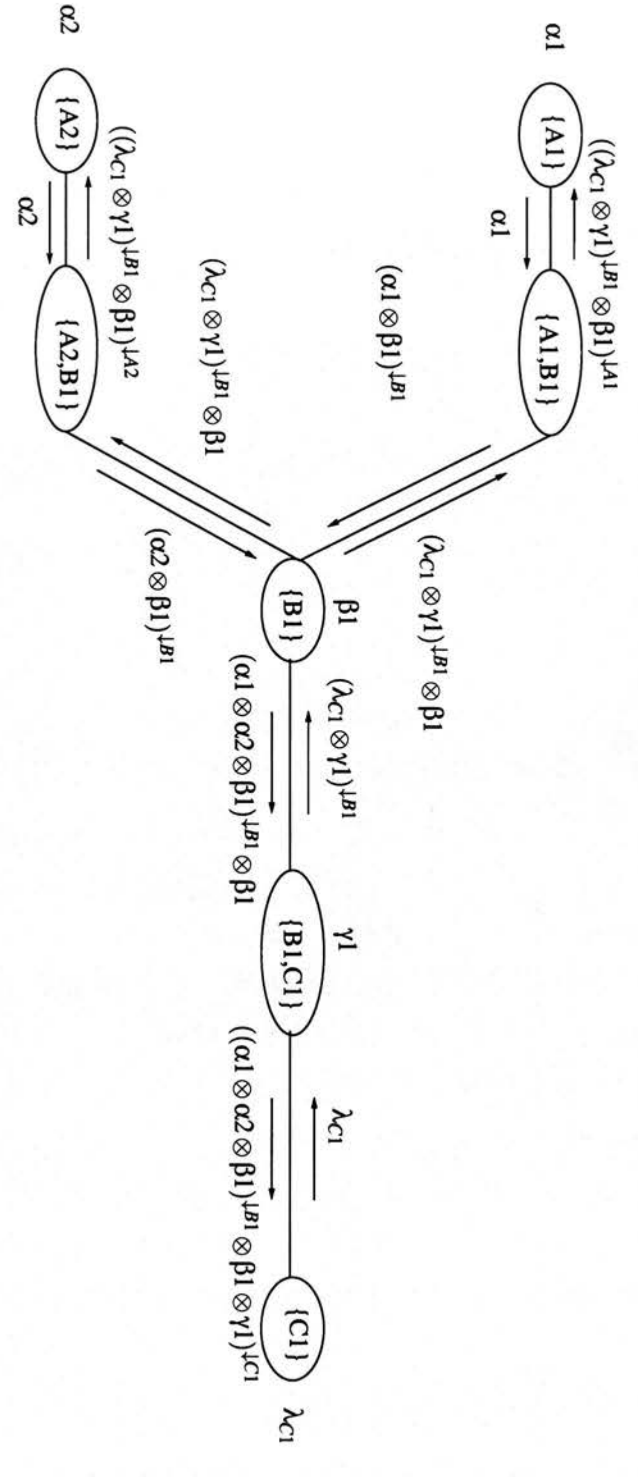

$\alpha1$
{A1}
$((\lambda_{C1} \otimes \gamma1)^{\downarrow B1} \otimes \beta1)^{\downarrow A1}$
$\alpha1$
{A1,B1}
$(\alpha1 \otimes \beta1)^{\downarrow B1}$
$(\lambda_{C1} \otimes \gamma1)^{\downarrow B1} \otimes \beta1$
$\alpha2$
{A2}
$((\lambda_{C1} \otimes \gamma1)^{\downarrow B1} \otimes \beta1)^{\downarrow A2}$
$\alpha2$
{A2,B1}
$(\lambda_{C1} \otimes \gamma1)^{\downarrow B1} \otimes \beta1$
$(\alpha2 \otimes \beta1)^{\downarrow B1}$
$\beta1$
{B1}
$(\lambda_{C1} \otimes \gamma1)^{\downarrow B1}$
$(\alpha1 \otimes \alpha2 \otimes \beta1)^{\downarrow B1} \otimes \beta1$
$\gamma1$
{B1,C1}
λ_{C1}
$((\alpha1 \otimes \alpha2 \otimes \beta1)^{\downarrow B1} \otimes \beta1 \otimes \gamma1)^{\downarrow C1}$
{C1}
λ_{C1}

Figure 2.6: The messages computed in the Shenoy-Shafer reasoning algorithm.

the information within its own distribution combined with the distributions of its message receiving neighbours, by using combination and marginalisation operators. The process starts when the nodes for which the marginals are to be computed request information from their neighbours, after evidence for a particular variable has been introduced. For the example network of Figure 2.5 the information sent between adjacent nodes, once the evidence is provided, are illustrated in Figure 2.6.

The final stage is known as the *marginal computation phase*. The marginals are computed during this stage for the desired subsets of the network variables using the following rule, (Shenoy and Shafer, 1988):

> When a node a has received a message from each of its neighbours, it combines the messages together with its own distribution v_a and reports the result as its marginal. If ϕ denotes the joint distribution, μ the messages received from the neighbouring nodes and N the set of neighbouring nodes, then
>
> $$\phi^{\downarrow a} = \otimes\{\mu^{t \rightarrow a} \mid t \in N\} \otimes v_a.$$

In the example network of Figure 2.5, for instance, the marginal joint posterior distribution for node C1 is

$$\lambda_{C1} \otimes \mu^{B1C1 \rightarrow C1} = \lambda_{C1} \otimes ((\alpha 1 \otimes \alpha 2 \otimes \beta 1)^{\downarrow B1 \otimes \beta 1 \otimes \gamma 1})^{\downarrow C1}.$$

The Shenoy-Shafer algorithm provides a consistent and formal methodology for handling any kind of uncertainty measure, e.g. probabilities, boolean certainties, possibilistic measures etc. It is selected for the implementation of the Bayesian network reasoning mechanism in this work on the basis of its flexibility and efficiency.

2.5 Summary and Contributions Overview

This chapter has provided an outline of the main research threads in the three areas of interest for the work described in this thesis: Automated Modelling with emphasis on Compositional Modelling techniques, Explanation Generation, Fuzzy Logic and Bayesian Networks. In particular, the discussion covered the following:

In relation to Compositional Modelling: Methodologies developed to address the needs of the different stages involved in automated modelling were described. Of particular importance to this thesis are the works that fall under the general category of advanced compositional modelling, namely **Compositional Modelling**, (Falkenheiner and Forbus, 1991), and its extensions: **Causal Approximations**, (Nayak, 1994; Nayak and Joskowicz, 1996), **Relevance Reasoning**, (Levy et al., 1997), and **Dynamic Constraint Satisfaction CM**, (Keppens, 2002). These methodologies provided the basis upon which to develop the model fragment selection mechanism that is described in detail in the following chapters (Chapters 3 and 4).

In relation to Explanation Generation: This review covered the main issues that are involved in the task of generating explanations. The discussion revolved around the definition of the explanation generation task, the determination of user needs for explanation content, techniques for the extraction of such content and the main methodologies for the generation of explanations based on the determined explanation content. Of particular importance to this thesis is the work in (Cawsey, 1991, 1993) which provides the main theoretical principles for the explanation generation module. This thesis does not aim to develop a novel methodology for the generation of explanations.

For Fuzzy Logic: Fuzzy logic is applied in this work for the mapping of the perceived user understanding and preferences for explanation detail to the required representation detail for domain objects that the user is interested in getting explanations for. Thus, this work does not aim to extend the existing methodologies for Fuzzy Logic in any way. Section 2.3 outlines the basic Fuzzy Reasoning principles that are utilised in this thesis.

With regard to Bayesian Networks: Section 2.4 overviews the main research threads regarding Bayesian networks which are of interest to the work described herein. The main concepts described are the construction of Bayesian networks, the elicitation of network probabilities, and the usage of Bayesian networks for inference. Of particular importance for this work are the methodologies for the propagation of evidence in a Bayesian network, and especially the Shenoy-Shafer Belief-Function propagation method, outlined in Section 2.4.3.2, which is the preferred method for propagation of evidence in this work.

The main theme of this thesis is the automated formulation of models for engineering domain systems in order to prepare human targeted explanations for these systems. Inherent to this task is the requirement to manage the uncertainty sources within the communication with the user and the interpretation of his/her actions during the course of the explanatory interaction. These uncertainty sources can be identified in:

- Determining the user's level of understanding, which can only be done approximately.
- Handling the non-deterministic task of specifying the proper representation complexity for sub-parts of the domain system of interest, and of propagating the effects of such a specification to determine the required representation for the remaining parts.
- Tackling the complex task of mapping all possible levels of the perceived by the program user understanding to all appropriate detail levels for the models that can be formulated.

Existing techniques of Explanation Generation, as summarised in Section 2.2, provide some methodologies for assessing the user's explanatory requirements, and performing some simple modelling the user's understanding during the explanatory interaction. However, the treatment of uncertainty usually remains simple, associating simple measures of modelling the user's understanding to the level of detail required in the generated explanations. As such, there is still much left to be done with regard to tackling the uncertainty that usually arises due to the approximative nature of user modelling. Furthermore, there is yet no comprehensive solution to the problem of handling the non-determinism of propagating the effects of preferring particular detail levels for some but not all of the composing parts of the domain system involved in an explanation generation context.

On the other hand, automated model formulation techniques, as discussed in Section 2.1, provide coverage for the variation of the detail of the models which are formulated for domain system representation. However, they usually lack the user-friendly communication, in the context of explanation generation. It is customary in existing CM research to require the user to communicate with the program using constraint satisfaction queries, typically in a form that is intricately associated with the type of formalism employed by the particular constraint satisfaction techniques that tackle the issues surrounding the formulation task.

This thesis aims at bridging these two worlds. By providing a link between existing explanation generation techniques and a model formulation engine, it aims at covering the need for a better communication between the user and the model formulation algorithms. By allowing the explanation generation front end to affect the formulation of model representations for the domain systems involved, it is possible to enable the variation of the detail of the explanation content based on user requirements, through varying the detail level of the representation used for content extraction. A fuzzy logic module aims at bridging the gap between assessing the user requirements based on his/her indicated preferences and providing evidence toward the selection of detail level for those parts of the domain system that are of interest. Thus, the aim of the fuzzy logic module is handling the complexity of mapping types of user understanding to acceptable levels of domain representation detail. Finally, the addition of a Bayesian network propagation engine enables the extension of the effects of selecting representation detail for some sub-parts of the domain system of interest to determining the required representation for the remaining parts. One additional benefit of using a Bayesian network evidence propagation engine is efficiency. Most constraint satisfaction based compositional modelling techniques, described in Section 2.1, suffer from performance penalties when handling large model fragment libraries (indeed most of the reported examples in the corresponding literature are of small scale). To some extent, these performance issues can be alleviated by the usage of Bayesian networks instead of constraint satisfaction algorithms, though some performance issues still remain as described in Chapter 5.

Thus, a framework emerges for automated model formulation based on approximate reasoning techniques to tackle explanation generation. The analysis of this framework will be the focus of the next chapter.

Chapter 3

Framework for Model-Based Explanation Generation

Models, as discussed in Section 2.1, are conceptual approximations of real-world systems which can facilitate focused examination of certain aspects of these systems.

The issues revolving around the modelling process are (Keppens and Shen, 2001):

- Representation formalism: a (symbolic) language that allows the representation of relevant aspects of the system being modelled and their relations. The formalism significantly affects the range of models that an automated modelling computer program can represent. Since models are approximate conceptual descriptions, the representation formalism can be designed in such a way as to help describe the variation of what a model represents according to different 'modelling dimensions', (Leitch et al., 1999) such as precision, resolution, scope, granularity etc.

- Problem: a task that must be solved with respect to the system, such as diagnosis, explanation, plan synthesis, monitoring or prediction.

- Problem solver: an inference procedure capable of generating one or more solutions for the problem given a description of the system in the appropriate representation formalism. As described in (Keppens and Shen, 2001) this procedure can be categorised effectively as deductive, inductive or hybrid.

Using this terminology, a model can be described as an instance of the representation formalism that helps demonstrate certain properties or behavioural aspects of the actual system that is being modelled, and that enables the problem solver to apply its inference procedures. Modelling as a process involves mainly collecting sufficient evidence for the system's aspects or behaviour to be demonstrated & performing inference based on this evidence and driven by the goal of the task at hand. The result can be expected to be a final proposition of a model (or set of models) which can fulfil the task requirements and which are valid when compared to the real-world system itself.

In an attempt to address the above three issues a framework is constructed to tackle a given task using a particular representation coupled with an inference procedure. In this chapter the framework for the present work is shown, offering an architecture for explanation generation which employs an automated model formulation engine that aims to provide knowledge representations (models) to support explanations for real-world systems (or processes) to users of varying degrees of expertise with regard to these systems.

3.1 Why is there a Need for this Framework?

The need for generating informative explanations regarding the structure and behaviour of physical systems plays a major role in the setting of industrial training. Automating the task of explanation generation is considered paramount to providing cost effective as well as customisable industrial training tools, though it introduces a set of "open" design questions:

- Range of training facilities provided.
- Broadness of domain coverage in terms of explanation content.
- Maintenance of the focus on the significant aspects that may be of importance to the particular user.

- Robustness of handling different explanation requests.

Although many methodologies and explanatory programs have been proposed so far (Cawsey, 1991; Suthers, 1993, 1994; Carenini and Moore, 1991), major challenges remain. For instance, natural language-based explanation generation facilities typically lack effective and modular knowledge representation facilities which are adaptable to the user expertise. On the other hand explanatory programs based on the Compositional Modelling (CM) paradigms (Gruber and Gautier, 1993; Nayak and Joskowicz, 1996) may provide flexible knowledge representation formalisms and inference methods, but they usually suffer from stiffness of the generated explanatory content and from absence of effective communication with the person requesting an explanation for a real-world system.

The framework proposed in this thesis exploits model-based reasoning techniques to support representational modularity as well as adaptation of the explanatory content. It addresses the need of performing systematic adjustments to the detail level of the models that are formulated for the domain systems which are the target of the required explanation generation, using the estimated expertise of the user as a guide. CM is used as the basis of implementing such adjustments. In particular, CM is extended with the employment of Bayesian networks reasoning, to enable the selection of appropriate fragments which would reflect the expertise level of the user.

Note that no claim is made that this reflection of the user requirements in the detail level of the formulated models is cognitively plausible. The automated model formulation process does not have to mirror human modelling. Human modellers use a great deal of tacit knowledge about the real-world system of interest. In this work, only information about the system of interest is utilised, which is made explicit via knowledge of the domain in conjunction with a heuristic determination of user expertise levels, in deciding how to associate such a level of expertise to the details of the formulated model that is explained to him/her. The usage of the formal Bayesian networks (Pearl, 1988; Lauritzen and Spiegelhalter, 1988; Shenoy, 1997) theory helps to formalise this association.

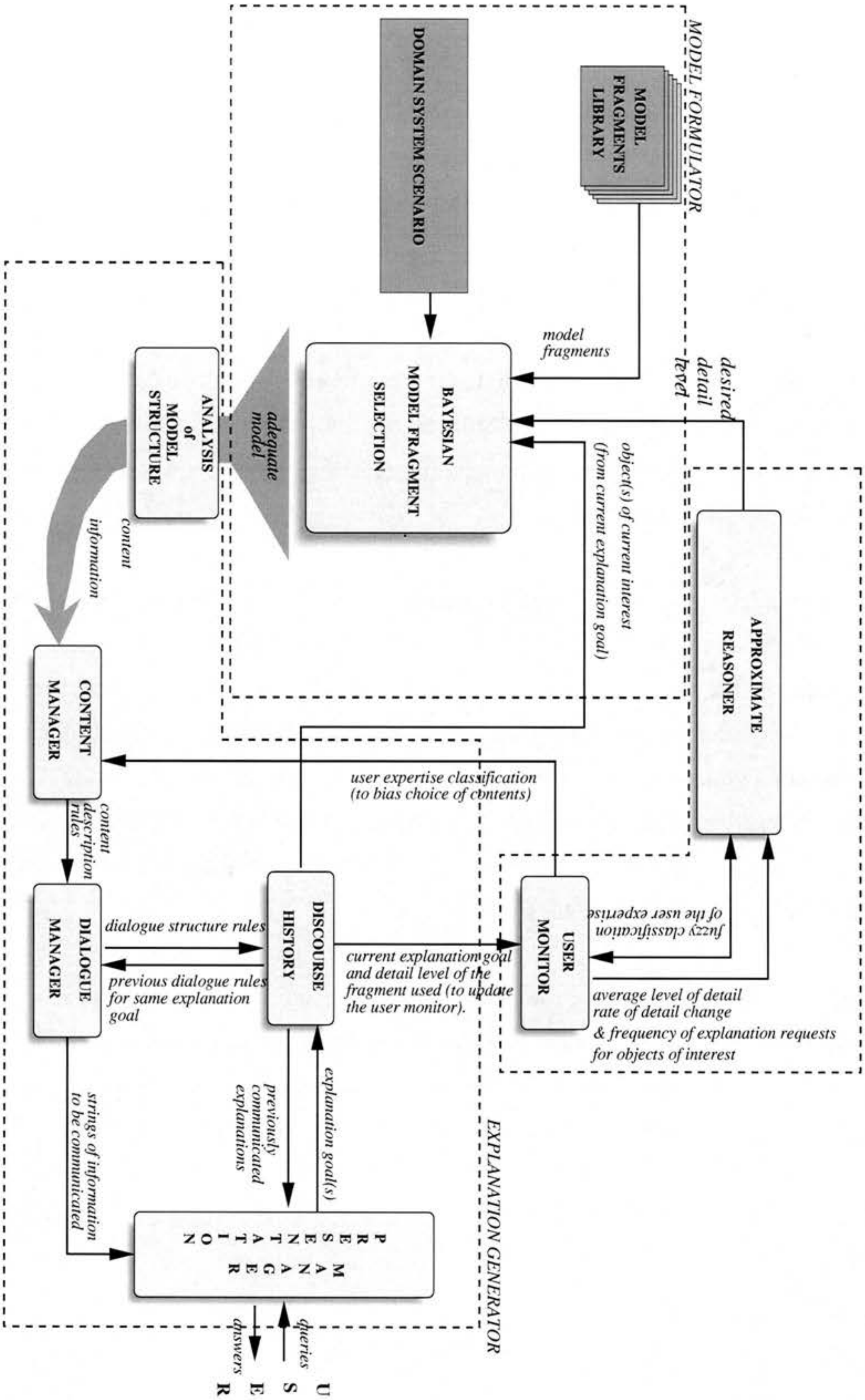

MODEL FORMULATOR
MODEL FRAGMENTS LIBRARY
DOMAIN SYSTEM SCENARIO
model fragments
desired detail level
BAYESIAN MODEL FRAGMENT SELECTION
object(s) of current interest (from current explanation goal)
adequate model
ANALYSIS of MODEL STRUCTURE
content information
APPROXIMATE USER MONITOR
APPROXIMATE REASONER
CONTENT MANAGER
user expertise classification (to bias choice of contents)
content description rules
fuzzy classification of the user expertise
USER MONITOR
average level of detail
rate of detail change
& frequency of explanation requests
for objects of interest
DIALOGUE MANAGER
dialogue structure rules
previous dialogue rules for same explanation goal
DISCOURSE HISTORY
current explanation goal and detail level of the fragment used (to update the user monitor).
strings of information to be communicated
previously communicated explanations
explanation goal(s)
PRESENTATION MANAGER
queries
answers
USER
EXPLANATION GENERATOR

Figure 3.1: Explanation framework: individual modules and their components.

3.2 Framework Architecture

The proposed overall framework for model-based explanation generation is shown in Figure 3.1. It consists of three modules: Explanation Generator, Approximate User Monitor and Model Formulator. The design specification for these modules, their interrelationships and the modules' component details are explained below. In particular, due to the significant role that domain knowledge modelling plays in model-based explanation generation, a major focus is given on the description of the Model Formulator.

3.2.1 Explanation Generator

The Explanation Generator serves as a mediator with the user. During a training session, the user selects questions from predefined menus about the components of the domain system. The questions that may be asked are of the types "what-is-it" (e.g. for the secondary liquid sodium loop domain, as described later in Section 3.4, "what is an Intermediate Heat Exchanger"), "how-it-works" (e.g. "how does an Intermediate Heat Exchanger work"). These types of questions may be used in order to extract explanations for the structure and/or behaviour of domain components as well as for the relationships between these components or between variables associated with component quantities.

For each question the user can either select a desired level of representation detail, thus having a way of controlling the amount of information that will be provided in the answer, or let the detail be decided by the Approximate User Monitor (discussed in the next sub-section). The latter is the default action. In the current implementation, the information detail options provided for the user to select from vary along with the question types. For example, a "what-is-it" type of question for a particular component can only have one designated level of detail, i.e. providing all possible information on the structural details of the component. However, a "how-it-works" question can be asked at 3 different complexity levels, reflecting all possible answer types: a general statement about a component's function, a more complex identification of all elements of the component's normal operation, or a list of the component's faulty behaviour modes. Each selected question is internally transformed into a first-order predicate structure which forms the current *user explanation goal*: `question_type([list_of_related_objects], < desired_answer_complexity >)`, where an object is a domain component or process of interest. The provision of the desired complexity to be used in satisfying a given explanation goal is optional, depending on whether or not the user actually selects a specific level of complexity.

The explanation goal is utilised to update the User Monitor and the discourse history. The latter is a record of all salient objects during the explanatory interaction. For an instance of the interaction, each salient object consists of the following items:

- The most recent explanation request, in the form of the top-most dialogue plan entity which was generated in order to satisfy that request.
- The detail level used in each fragment of the model that was formulated to support the explanation generated for that request. A fragment's detail level is defined, in this work, as the fragment's resolution (Leitch et al., 1999), i.e. the number of endogenous variables defined in the model fragment. Other measures can also be employed, such as the precision of the values of the model variables, or the number of the model constituent independent modelling units (which is also known as "granularity" of the model). Further to this observation, it is plausible that human modellers use a combination of criteria to determine the complexity of a model. Nevertheless, the number of variables defined inside a modelling entity, such as a model fragment, has proven to be a quite effective measure of a model's complexity while being simple to use (Rickel and Porter, 1997).

The history of discourse serves a twofold purpose. First, it acts as the reference to previous instances of the interaction in order to discover important properties such as the frequency that an object has appeared within explanation goals and the detail level that the object was last modelled by the Model Formulator (see later). These detail levels are crucial in reasoning about the user's expertise, as well as in identifying what should be the detail level of the representation of a particular component in the subsequent models of the domain system. Second, it provides an essential means to guide the focus of attention of the user as the explanatory interaction continues. This is achieved by prioritising explanation goals in a hierarchical manner, using "preference" heuristics (Suthers, 1993), such as "saying something new" or "preferring satisfaction of long term goals rather than short term pursuits".

The Content and Dialogue Managers are used respectively for the generation of explanation content and that of dialogue structure. Once the domain system model has been formulated it is passed back to the Content Manager where suitable explanation content is extracted, in order to satisfy the explanation goal. The Content Manager uses heuristics, based on the perceived user understanding and the type of question asked, to identify semantic units known as "views" which outline the content of the explanation to be communicated to the user, with respect to a specific domain component (Cawsey, 1993) (more information about these heuristics is provided in Section 4.4). These views depend upon the scope of the model fragments used in the formulated model. For example, the component condenser for a typical Rankine cycle in a vapour power plant (Moran and Shapiro, 1992) can be *viewed* in the lowest scope level through the condenser's structural description, in larger detail through its function specification for normal operation, or in even larger detail through its behavioural diversity, covering both its normal and also faulty operation modes. Since a fragment's scope is controlled, in this work, by its resolution level these views reflect the model fragments used.

In conventional explanation generation programs, such views have to be encoded explicitly before producing any explanations, for all the domain components that could be of interest to the user. Specific methods have to be developed to retrieve the views that fit the programs' perception of expertise for a particular user, in order to generate the explanation content (Lester and Porter, 1997). In the framework proposed herein, the model fragments themselves provide the alternative views. This can simplify the process of designing a content generation module, to be just the definition of alternative methods for parsing the information encoded in a model's fragments.

The discourse structuring mechanism essentially follows the lines proposed in (Cawsey, 1993). Since the interaction between the program and the user is in dialogue form, the structure is generated incrementally, using a hierarchical arrangement of the dialogue goals. An excerpt of such content generation plans is illustrated in Figure 3.2 for a "how-it-works" type of information request.

3.2.2 Approximate User Monitor

This is a tool composed of the User Monitor and the Approximate Reasoner, which estimates the expertise of the user. Its input consists of the detail variations in the representation of the objects appearing in the models formulated for previous communication instances and the level of the user expertise perceived so far. As output this module produces an estimation of the following:

- A suitable detail level for the fragments of domain objects. This is determined for those objects that appeared in models used in preceding explanations, as well as for the object(s) appearing in the current explanation goal if the user provided no desirable level of complexity for its (their) representation.
- Revised user expertise for the objects whose details were determined previously by the Reasoner.

The Approximate User Monitor obtains its input by evaluating information stored in the so called "profiles" of the User Monitor. For each explanation request the User Monitor creates a set of "profiles" for the user. Each profile is associated with one of the objects that appeared in the explanation goal for the user query when the query was posed. It is labelled with that object's name and the monitor's previous assessment for the user expertise regarding the particular object (if no explanation has been provided yet for the object then a default assessment can be provided). Within such profiles the Monitor records the explanation goal, and the resolution level of the fragment that was used to represent the domain object, which appears in the profile label, in the model which had been formulated to support the explanation already generated for that goal.

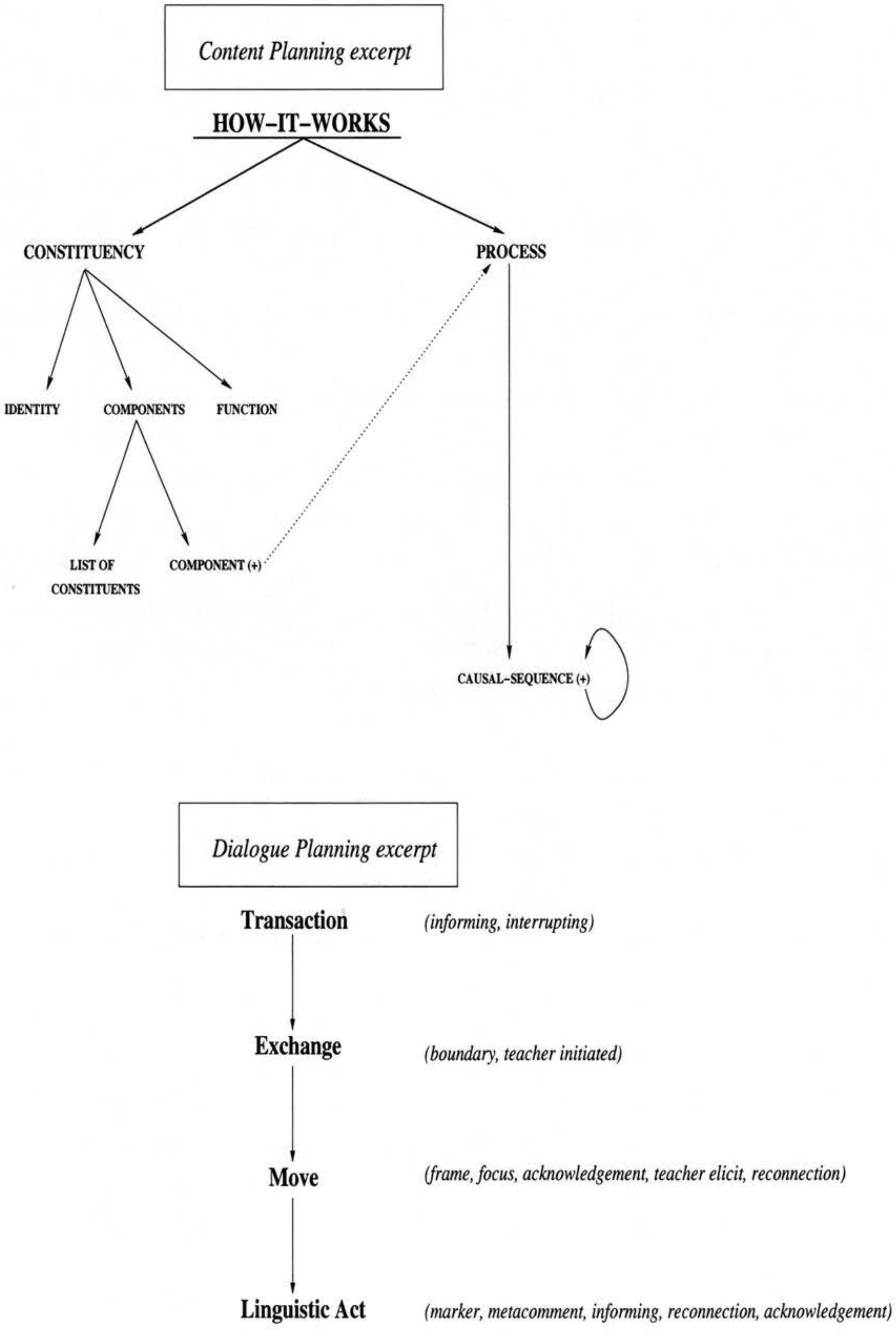

Content Planning excerpt
HOW-IT-WORKS
CONSTITUENCY
PROCESS
IDENTITY
COMPONENTS
FUNCTION
LIST OF CONSTITUENTS
COMPONENT (+)
CAUSAL-SEQUENCE (+)
Dialogue Planning excerpt
Transaction
(informing, interrupting)
Exchange
(boundary, teacher initiated)
Move
(frame, focus, acknowledgement, teacher elicit, reconnection)
Linguistic Act
(marker, metacomment, informing, reconnection, acknowledgement)

Figure 3.2: Excerpts of the plans generated for content and dialogue structure upon a user information request of the type "how-it-works".

From the information stored in these profiles, for a given object $object_i$, the average detail level (resolution) $\bar{R}_{\text{object}_i}$ of its representations in past explanations can be computed, as well as its rate of change of the resolution level $\dot{R}_{\text{object}_i}$ in the most recent representation of the same object, as the explanation process continues. These two measures, together with the frequency of occurrence of an object within all the profiles, F_{object_i} and the perceived user expertise level for the particular object $UEXP_{\text{object}_i}$, are used by the Approximate User Monitor to compute the *desired resolution level*, $\hat{R}_{\text{object}_i}$, of the fragments to be selected in the subsequent model formulation process. $\bar{R}_{\text{object}_i}$ together with $\dot{R}_{\text{object}_i}$ provide some means to determine how the user has been interacting with the program so far to acquire explanations for $object_i$. However, these measures alone cannot provide a good picture of the interaction the user has had. To achieve a better view of the interest that he/she has shown in $object_i$ it is important to determine the frequency of requests for explanations for that object, via F_{object_i}. This measure provides an intuitive association of more frequent requests for information to a desire to find out more due to lack of understanding. $UEXP_{\text{object}_i}$, finally, provides some counterbalance to the possibility of misjudging an inquisitive expert for someone that may have trouble in perceiving the explanations given.

Although the three measures mentioned above, $\bar{R}_{\text{object}_i}$, $\dot{R}_{\text{object}_i}$ and F_{object_i} are themselves not fuzzy, rules regarding the associations between them and both the desired detail for the model fragments and the user expertise level are often given in linguistic terms. Therefore, the employment of Fuzzy Reasoning (Pedrycz and Gomide, 1998) in the present work provides a useful mechanism to account for the uncertainty that accompanies such associations.

Obviously, fuzzification (Pedrycz and Gomide, 1998) is required prior to any reasoning process, mapping the given crisp input values onto fuzzy values. In the present work, the input and output variables may take their fuzzy values from the following quantity spaces (with different underlying semantics of course). The following definitions of the fuzzy value spaces serve for the thesis in demonstrating the main ideas described herein, though they do not impose any limit to the generality of the approach. It is envisaged that ultimately these spaces can be configurable by the user.

- $F_{\text{object}_i} : \{zero, low, medium, high\}$.
- $\bar{R}_{\text{object}_i} : \{low, medium, high\}$.
- $\dot{R}_{\text{object}_i} : \{negative, zero, positive\}$.
- $\hat{R}_{\text{object}_i} : \{low, medium, high\}$.
- $UEXP_{\text{object}_i} : \{novice, intermediate, expert\}$.

It should be noted that the fuzzy values of *zero* for the variable F_{object_i} and all the values of variable $UEXP_{\text{object}_i}$ are fuzzy singleton sets, i.e. each has a single member with membership 1.

The general structure of rules used in the Reasoner is shown in Figure 3.3. However, a different set of rules applies for each type of question complexity (question complexity is discussed in Section 3.2.1). By selecting the complexity, which the program should consider when formulating a domain representation to use for an explanation, a user implicitly selects also a particular rule-set to be used by the fuzzy reasoner. This allows a finer control of the inference performed by the fuzzy reasoner with respect to different complexity levels. An example of the structure of the rules in table format is given in Table 3.1 (detailed description of the entire rule-set is given in Appendix A). The table provides an overview of the subset of rules when $UEXP =$ Novice, and $\bar{R} =$ Low, for all cases of selection of the complexity of the explanation that the user seeks. All rules have been determined empirically.

Given a selected rule-set and the actual measurement of the input variables, the Approximate User Monitor computes the desired resolution level and the user expertise by firing the rules matching the fuzzified inputs. The resulting value of the resolution level is in general a weighted combination of the fuzzy values *low*, *medium* or *high* and similarly the value of user expertise is a weighted combination of the fuzzy values *novice*, *intermediate* and *expert*. The resulting value for the user expertise variable is used for labelling the profile(s) that the Monitor creates for the current explanation goal. A final step in the function of the fuzzy reasoner involves defuzzification (Pedrycz and Gomide, 1998) of the two weighted combinations resulting in crisp values for both output variables. The defuzzified value for user expertise is used to bias

$$
\begin{aligned}
\textbf{IF } \bar{R}_{\text{object}_i} \; is & \; < \text{Fuzzy Value} > \quad AND \\
\dot{R}_{\text{object}_i} \; is & \; < \text{Fuzzy Value} > \quad AND \\
F_{\text{object}_i} \; is & \; < \text{Fuzzy Value} > \quad AND \\
UEXP_{\text{object}_i} \; is & \; < \text{Fuzzy Value} > \\
\\
\textbf{THEN } \hat{R}_{\text{object}_i} \; is & \; < \text{Fuzzy Value} > \quad AND \\
UEXP \; is & \; < \text{Fuzzy Value} >
\end{aligned}
$$

Figure 3.3: General structure of the rules used by the Approximate User Monitor.

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
L	Z	Z	Novice	L	Novice	M	Novice	H	Novice	L	Novice
L	N	L	Novice								
L	Z	L	Novice								
L	P	L	Novice								
L	N	M	Novice	L	Novice	M	Interm	H	Interm	L	Interm
L	Z	M	Novice							L	Interm
L	P	M	Novice							M	Interm
L	N	H	Novice	L	Interm	M	Interm	H	Expert	M	Expert
L	Z	H	Novice								
L	P	H	Novice								

Table 3.1: Fuzzy rule-set for $\bar{R} =$ Low and $UEXP =$ Novice

the Content Manager's selection of proper parsing facilities when content is extracted from a formulated model.

The User Monitor is not an attempt to provide a complete model of the user's expertise. Instead, this module provides the means for "controlling" future detail level of the domain models based on past detail values and on frequency of reference to a particular domain system component. Although there are more elaborate techniques for modelling users, a consistent and robust user model is usually hard to obtain. The trade-off between the robustness, the effectiveness and the usability of a user model often causes debate as to how useful such a model could be (Cawsey, 1993; Moore, 1995). The present fuzzy reasoner offers a relatively simple but still plausible user monitoring scheme.

Once the representational levels of detail for at least some of the domain system's components have been decided, such decision along with the explanation goal and other important information is given as input to the Model Formulator, in order to select appropriate fragments and assemble those that are selected into an overall domain model.

3.2.3 Model Formulator

The Model Formulator exploits the fuzzy reasoner's decision to select appropriate model fragments. This module forms a main contribution of the present work. A compositional modeller, displayed in Figure 3.1 as "Model Formulator", for users with different levels of experience is exploited. The inputs to the Model Formulator are:

- A library of model fragments, representing the domain system components at different detail levels.
- A structural description of the domain system, specifying the components involved and the inter-connections between them.
- A set of objects (components or processes) of interest, obtained from the queries of the user.
- A metric of the user's desired level of detail.

Classical CM defines the model fragments as *abstract* descriptions of physical phenomena, containing only references to component types that can participate in these descriptions. Thus CM traditionally allows fragments to be "instantiated" once the components, which participate in a given simulation environment, are determined. However, for the task of explaining the functionality and/or behaviour components to users it may be more natural for the implementation of a fragments library to use fragments representing individual components that participate in a given domain system. Thus, in this work, each model fragment may represent structural, functional and behavioural properties of a particular component, depending on its resolution.

Fragments of the lowest resolution, which are referred to as simple fragments hereafter, contain the structural information for the modelled component, the most significant variables involved and the typical qualitative constraints that describe the component's normal behaviour. Fragments of intermediate resolution encapsulate simple fragments and additional knowledge for the complete functional description of the component's normal behaviour. High resolution fragments provide the same information as intermediate ones plus several alternative operating modes of the component (normal and faulty).

Different model fragments for a component are organised under an *assumption class* for that component. In traditional CM, fragments within an assumption class are deemed to have different and often mutually contradictory conditions. This is a constraint normally used to avoid inclusion of two alternative descriptions of the same phenomenon in the same composed model. Here, an assumption specifies a general condition under which the fragment applies and an assumption class groups together those assumptions relevant to a component. Since the Bayesian network (Pearl, 1988) fragment selection mechanism provides the means to automatically select only one fragment from the assumption class of each component, the restriction for mutually exclusive conditions among fragments of a single assumption class can be alleviated. This allows different model fragments to be defined within one assumption class whose application conditions are no longer mutually contradictory. For instance, an overlapping of conditions may occur between a high and an intermediate resolution fragments. High resolution, or complex, fragments are targeted especially at users who are perceived to be experts regarding the modelled phenomena, providing such users with an opportunity to identify alternative behaviour modes of a specific component. This offers a novel approach to the handling of assumption classes in CM.

The required structural description of the domain system is prescribed only once as part of the initial setting of the explanatory session, and remains unchanged during the explanation process for the domain of interest. The domain to be modelled is defined prior to model formulation. This is done by giving the initial description of the system that the trainee is to learn about, and deciding which kind of phenomena should be represented by the model (e.g. mechanical, electrical, hydraulic, etc).

The last two inputs, i.e. set of interesting objects and metric of user's desired detail level, are produced by the explanation generator, which interfaces with the user and by the fuzzy reasoning module respectively. Model fragment selection is guided by a Bayesian network that relates all applicable model fragments to one another. This Bayesian network enables the modeller to deal with uncertainty about the level of detail of the selected model and its constituent fragments that best suit the user's needs and preferences. Hence, this work illustrates how the symbolic methods of finding consistent models can be enriched with a numeric method to deal with the uncertainty that characterises the appropriateness of individual model fragments.

The first step of model fragment selection consists of the construction of the Bayesian Network. Each node in the network represents the selection or rejection of an individual model fragment instance. The structure of the network is based on the topological structure of the system being modelled. This is possible for training purposes, since the knowledge of domain systems' structure is well maintained. Each of the model fragments may represent the structural or behavioural description of a component or process under certain operational conditions and is associated with several potential consequences that may hold if the conditions are satisfied. Different model fragments, modelling the same component, are grouped in assumption classes, strictly ordered within each assumption class with respect to their level of detail. When the outputs of a component are connected to the inputs of another component, their respective model fragments are linked to one another. Domain heuristics are utilised to assign probabilities to the root nodes and the links required by the modelling network.

The model fragment selection process amounts to selecting the representational complexity of the model's subparts within the specified domain, via selecting the resolution (Leitch et al., 1999) of the available fragments. The outcome of this reasoning is a complete model, i.e. a model which contains one model fragment for each component of interest and which is appropriate for the knowledge level of the trainee. The employment of Fuzzy Reasoning techniques in the User Monitor and the exploitation of the Bayesian Network also help in selecting a set of model fragments that are individually deemed to best suit the task at hand, explaining the domain system to a user with particular expertise requirements. Thus the resulting final model is both complete and adequate since the individual components are represented adequately in it. The model is parsed in order to extract information to be conveyed to the user via the front-end (shown as "Presentation Manager" in Figure 3.1). The final front end can be envisaged as simple or as complex as the reader desires. For the purposes of demonstrating the ideas in this thesis however, it suffices to assume it as simple as a text-based interface, without incurring loss of generality of the core reasoning approach related to model formulation. In such an implementation, the front-end should allow the user to provide input by text-based selection of the following:

- The type of the question asked

- The object of interest for that question

- The desired detail level for the explanation that will be generated. This can be set either by selecting a detail level from a list of acceptable levels (1 - 3, 3 standing for the highest available detail level), or let the program determine automatically the optimal detail level for the particular user, based on previous interaction.

The interface envisaged would provide the final textual explanation to the user, once the model formulation and explanation content generation are completed.

3.3 Perspective: How the Framework Addresses CM-related Issues

In general, the approach of the framework summarised in the previous section has similarities with the compositional modellers as discussed in Chapter 2, except for the explicit use of numeric uncertainty to reason about model selection. Thus, it shares by definition some of the main strengths associated with CM:

- Generality across domains: the representation formalism used does not necessarily limit the scope of the potential domains that the framework can be applied to. However, the heuristics used in the present work to determine the structure and probabilities of the Bayesian network are closely associated with the task of explanation generation to meet the varying user requirements. Yet, these heuristics can be redefined according to the need for particular tasks and are not necessarily domain dependent.
- Specificity to the explanation generation task: the heuristics associated with the construction of the Bayesian network are particularly suited for the task of explanation generation. Tacit knowledge associated with particular domains can also be exploited to alter the inner workings of these heuristics, thus enabling the use of more specific assumptions to drive the Bayesian network inference mechanism.
- Modularity: the specific applications of composable fragments are not required to be known at the time of their definition as the knowledge base of the Model Formulator that consists of such fragments reflects only theoretical knowledge regarding a domain. Each fragment is an independent definition of specific processes related to a component of the domain. In addition to making the maintenance of the knowledge base manageable, this modular structure allows the modelling of a large range of systems within the particular domain. Furthermore, this modularity facilitates the description of the domain phenomena and processes from different perspectives, by varying the fragment properties that determine how general, accurate, complex this description may be. Thus modu-

larity of representation enables elaboration on the represented phenomena across variable modelling dimensions, such as resolution, generality, scope, accuracy and others, (Leitch et al., 1999).

In extension, the approach described in this thesis introduces an alternative methodology for selection of appropriate fragments such that the expertise level of the user would be accounted for in the representational detail of the fragments. The reasoning that is performed during the selection process, in the context of explanation generation, faces the requirement to cope with various sources of uncertainty:

- The determination of a user's level of understanding can only be approximate, since it is based on a prescribed set of abstract principles (such as those laid out in (Paris, 1989; Cawsey, 1993)) which, although general, cannot be deemed as fitting exactly each user's personality.
- The specification of what is "proper complexity" for the representation of the model's subparts, in order to satisfy the user's information demands and comply to the system's beliefs about the user's understanding, can only be determined in a vague manner. In a dynamic interaction, with the user's information demands and understanding altering the explanation context, this vagueness increases. At best, it is possible to identify the most suitable representation for some of the domain system components, based on the complexity of fragments used for these components in previous interaction instances and the current explanation context. To complete the model it is necessary to be able to *propagate* the effects of selecting some fragments with an amount of certainty, throughout all of the remaining components' available fragments, and attempt to identify suitable representations for these components.
- The mapping of all possible levels of user understanding to all possible component representations is a complex task. Due to this complexity, an exhaustive mapping is neither feasible nor desirable. On the other hand, using characteristics of the model fragments such as their resolution or precision to identify their suitability for different levels of user understanding can alleviate much of the complexity, but causes non-deterministic results when these mapping instances

mix together.

A practically useful model formulation mechanism should be able to handle these issues. The combined usage of Fuzzy Reasoning (FR) (Pedrycz and Gomide, 1998) and Bayesian network (BN) inference mechanisms, such as those described in (Pearl, 1988; Shenoy and Shafer, 1988; Shenoy, 1997), provides a flexible solution for addressing the aforementioned uncertainty factors. Specifically, the usage of Fuzzy Reasoning (FR) theory is motivated by its means for handling the vagueness in identifying the user's understanding using approximate (fuzzy) labels for the various characterisations of the understanding level. Furthermore, FR techniques offer a powerful scheme for combining mappings of the understanding levels to the representational complexity of available model fragments, thus addressing the third of the above bullet-list issues. BN reasoning mechanisms, on the other hand, handle efficiently the complexity of propagating the effects of selecting fragments for some of the domain system components throughout the set of the available fragments for the remaining components. In addition, BN theory provides an intuitive vehicle to represent the relationship between model fragments with respect to how (and how much) they can influence each other's selection.

However, the framework is also bound by some important limitations of both CM and Bayesian networks technologies, most importantly from the ones that are described below:

- Knowledge acquisition. This is characterised as the most significant bottleneck in knowledge-based systems research (Luger and Stubblefield, 1993). Although the knowledge engineer that exploits Compositional Modelling is not required to foresee all possible combinations of components and sub-components that may be used within a particular domain system, he/she is required nevertheless to provide or otherwise obtain complete and coherent descriptions of the individual components within assumption classes. In the present work this also entails providing descriptions for possible failures of corresponding components within fragments with higher resolution.

  The current work faces a two-level difficulty: in representing domain systems and their associated processes via fragments for CM, and in eliciting the structure and probabilities for the Bayesian networks. In as much as the work in this thesis is concerned, the elicitation of probabilistic measures has been automated to a certain degree via the use of domain specific heuristics that will be detailed in Section 4.2.2. The elicitation of the Bayesian network structure, which is simplified for the demonstration purposes in this thesis to complete connectivity of all nodes between adjacent assumption classes in the network, can also be automated using similar domain specific heuristics. However, both are relying on the acquisition of the actual phenomena representations encapsulated by model fragments within the physical domain of interest. Without these representations the generated models would be of no value for the purpose of generating explanations for the particular domain.

Advances in the area of ontologies (Neches et al., 1991) have brought on some progress in the sharing of concepts and model fragments or fragment libraries between researchers. In this model of knowledge representation, a library can be populated incrementally, using the same agreed language for notation of systems and phenomena for various physical domains, hoping in the simplification of the exchange of ontological data for these domains between research groups and organisations. However the efficiency of this knowledge sharing ability as provided by current Ontology technologies has been recently questioned (da Silva et al., 2002), thus allowing scepticism on the sufficiency of these technologies when adopted as medium for representation in Compositional Modelling.

- Task specification, i.e. formulation tailored to specific task targets such as explanation generation for the framework presented herein. In theory, the models constructed from the representations of the model fragment library are generic declarative descriptions of what is known for the domain system which is being modelled. Correspondingly, the model formulation mechanism can theoretically be a generic problem solver that does not tie closely to the task at hand, be that explanation generation, diagnosis etc. In practise, however, task specific assumptions and design considerations regarding the formulation algorithms are taken into account to provide more efficient or more accurate (in terms of using fragments that describe the relevant domain phenomena with greater accuracy). For the work described in this thesis, the assumptions that are taken into account to facilitate the model formulation process are mostly related to the structure of the formulator's Bayesian network as well as the heuristics used to allocate probabilities to the network, as described in Section 4.2.1 and 4.2.2.

- Completeness and adequacy assumptions. Complicating further some of the acquisition of model fragment representations the finally formulated models must also be coherent (i.e. they should include model fragments only for the *relevant* parts of the modelled system, depending on the task at hand) as well as complete (i.e. they should include fragments for *all* relevant parts of the domain system). The constraints that need to be satisfied for a formulated model to be considered adequate are normally determined by the task at hand, and are largely quantifiable around the already mentioned modelling dimensions (Leitch et al., 1999). These constraints test the final model accuracy/completeness (checking whether the relevant variables are included or whether the required domain system components or phenomena are represented within the scope specifications for the model), and coherence (checking whether the model includes any representations for components which are not directly associated with the current task).

  Certain compositional modelling techniques attempt to alleviate this issue by verifying each composed model against a set of constraints that test its adequacy for the given task (Nayak and Joskowicz, 1996; Rickel and Porter, 1997; Keppens, 2002). An alternative, and possibly more principled, approach would be to verify and validate the model fragment library in its totality before doing any model construction. Though there is no actual developed research work that tackles this latter approach, in this framework it is assumed that the model fragment library is verified for its suitability to the formulation of models for the particular domain system prior to the actual formulation taking place. Based on this assumption the final formulated model is *adequate* because all model fragments used are free of incoherent phenomena descriptions and *complete* since the fragment selection mechanism uses one fragment for each of the components in the domain system.

- Model formulation optimisation. In mainstream CM, both the model fragment selection and the model composition are inherently intractable tasks. Only a few CM algorithms, targeted mainly for use in time critical tasks such as diagnosis and control or monitoring, attempt optimisations such as trying to finish in polynomial time with respect to the number of model fragments made available to the selection process. Other optimisation techniques may be feasible, most notably via the employment of Genetic Algorithm optimisation methodologies (Goldberg, 1989). In any case, for the work described in this thesis the source of complexity can be identified to be the complexity inherent in the Bayesian network inference mechanism.

  In particular, the complexity pertains to the variable elimination algorithm that is central to the valuation-based inference mechanism used for this work. Based on results given in (Dechter and El Fattah, 2001) the primary interest for the complexity of the valuation-based inference algorithm adopted in this work is dependent upon the tree-width of the network used. Specifically, the time complexity is linear in the size of the variables (i.e. the size of the set of permissible values for each variable in the network), and exponential in the tree-width, which for each node in the network is defined as the number of the earlier neighbours to which the node is connected (Dechter and El Fattah, 2001). This determines that when inference speed is of importance to the explanation generation task the connectivity of each node to other nodes should be kept at a minimum, whereas the size of the set of values attributed to each node is of no significant importance. This matter is discussed further in Chapter 5.

3.4 Example Scenario of Model Formulation

To demonstrate the reasoning involved in the framework that has been described so far for model-based explanation generation, a simple domain system such as the one shown in Figure 3.4 will be used throughout this thesis. It is a simplified version of the secondary liquid sodium cooling system, a typical cooling system for fast breeder nuclear power plants. Sodium is the coolant of choice since it has poor neutron moderating abilities and terrific heat transfer characteristics. Being a highly conductive liquid metal most of the heat transfer occurs through conduction. Sodium does, however, absorb some neutrons and transforms into Na-24 which is radioactive. The secondary sodium loop ensures that no radioactive sodium leaves the reactor vessel. The entire coolant system is unpressurised since sodium is liquid at atmospheric pressure, thus minimising the danger of explosive loss of coolant from transfer pipes. The entire system works as a power generator via a steam generator (or evaporator as illustrated in Figure 3.4) which uses the heat transferred from the main reactor to produce steam which is fed into a power generating turbine (not shown in the simplified version of the illustrated system). The rationale behind selecting this system for the illustration of the framework reasoning can be attributed to this system's simplicity, since it contains several components connected in sequence without feedback loops, and also to the fact that the physical phenomena involved with this system are well studied. Despite its simplicity, it is a domain system sufficiently large to be a realistic choice of a real world system that can be handled by the framework.

Seven components appear in this illustration: IHX (intermediate heat-exchanger), P/M (pump-motor system), EV (evaporator), SE (source of liquid sodium), R1, R2, and R3 (hydraulic resistances of the connecting pipes). For simplicity only basic hydraulic and thermodynamic phenomena will be considered in the representation of these components for the automated model formulation process.

The library of model fragments for this example system contains 3 fragments for each component, 1 fragment per resolution level. Each component's fragments are grouped within the component's assumption class. The resolution levels of the fragments used for the individual components can vary depending on the domain. For this example the resolutions of the fragments are shown in Figure 3.4 as triplets of numbers next to each depicted component. For the purposes of this example the fragments are arranged so that there is one fragment per resolution level in each assumption class, though it is possible to have multiple fragments per resolution level within an assumption class (details for the implications arising with such a design for the model fragments library

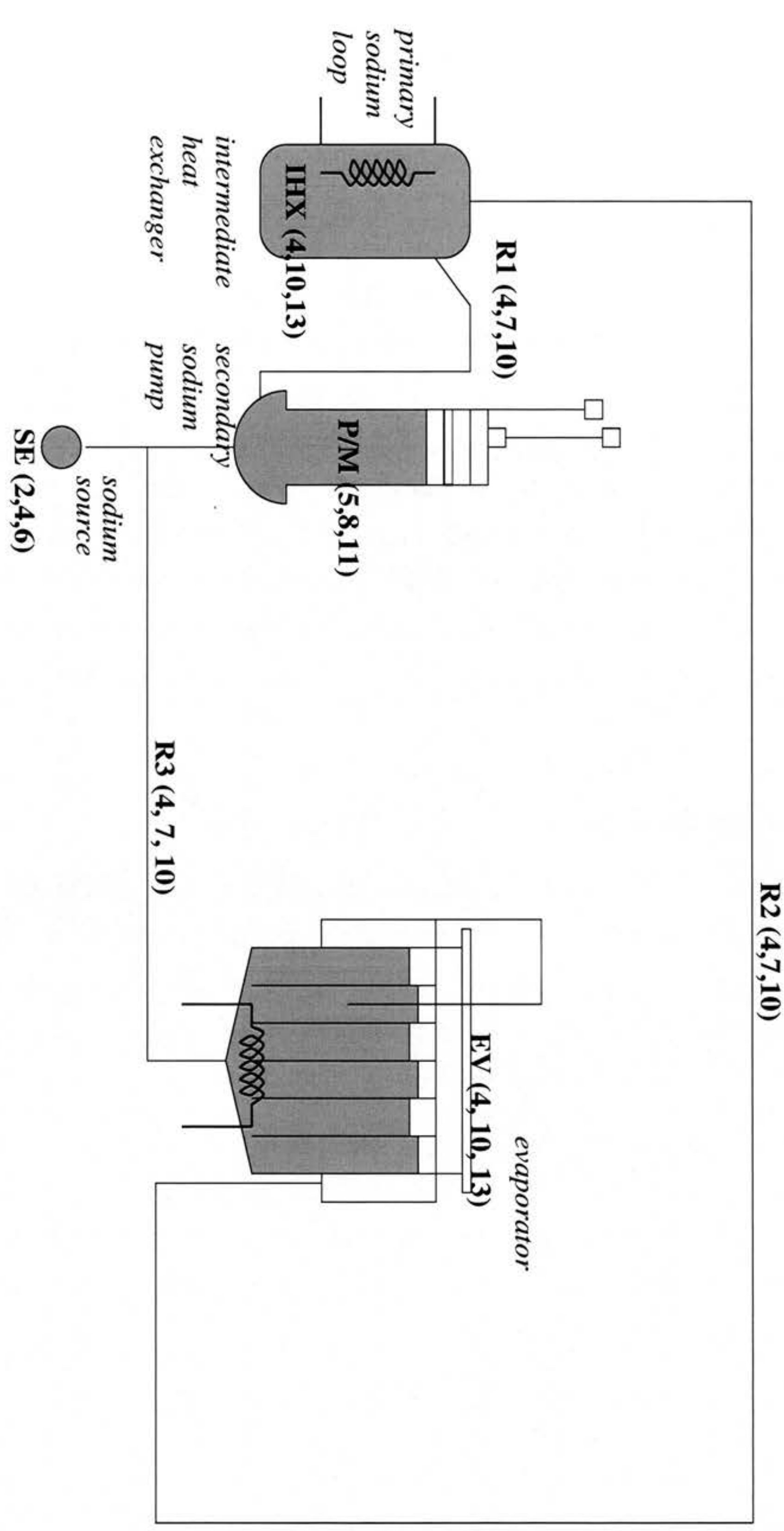


Figure 3.4: Liquid sodium cooling loop.

are discussed in Section 4.1.1. For this domain system a component is connected to other components via "terminal" variables: the liquid sodium flow and pressure.

Within this initial representation of the given domain system two explanatory interaction scenarios are explored in subsequent chapters:

- A "novice" user requires information about a particular component at the lowest possible detail level. The model formulation mechanism selects a suitable fragment for each of the remaining components in the domain system and prepares the model which is used by the Content Manager and Discourse Manager modules to generate the explanation body. This scenario focuses on outlining the function of the content and discourse generation modules.

- A user with different achieved levels of expertise for certain components visits more than one component in a particular explanatory session (i.e. following a series of interactions for more than one component with the explanatory program). The user's level of expertise varies during the interactions. This scenario is mainly targeted at demonstrating the working of the Model Formulator together with the fuzzy reasoner, while putting the functioning of the remaining framework modules in the context of varying the component representations.

Chapter 4

Development of the Model Formulator

The Model Formulator, briefly introduced in Section 3.2.3 forms the core of the Approximate Model Composition Framework. It handles uncertainty in selecting fragments with a sufficient level of representational detail to meet user explanation requirements. This chapter focuses on describing the requirements and work of this particular module of the framework. At the same time it provides a description of how this module connects to the Approximate User monitor and the Explanation Generation modules, the two other main modules of framework.

The design and implementation of the Model Formulator is demarcated by the following novel characteristics:

- A method for the selection of suitable fragments, based on the use of a combination of Approximate Reasoning methodologies and Bayesian network inference techniques to identify fragments that suit user requirements.
- An apparatus of representing model fragments using XML in order to tightly control the acceptable grammar for the definition of model fragments.
- A representation of rules for the generation of explanation content and dialogue structure, again using XML, to allow the binding of the information carried by the model fragments to a systematic way of presenting that information to the user.

4.1 Inputs to the Model Formulation Process

As indicated in Section 3.2.3 the Model Formulator receives the following inputs prior to any reasoning process of its own:

- A library of model fragments, representing the domain system components at different detail levels.
- A structural description of the domain system, specifying the components involved and the inter-connections between them. This is also known as the scenario regarding the domain system of interest. It is used only at the initialisation, to assist the structuring of the Bayesian network used by the Model Formulator, as will be described later.
- A set of objects (components or processes) of interest, obtained from the queries of the user.
- A metric of the user's desired level of detail.

The following subsections will elaborate on these inputs using the example of Section 3.4 as a guide. A discussion of some of the most important issues regarding the selection of model fragments will follow the description.

Notationally, a *model* is herein represented as a pair $\langle O, C \rangle$ of object constants $O = \{o_1, \ldots, o\mathrm{v}\}$, which for this work stand for the components of the domain system, and relations over these object constants $C = \{c_1(o_1, \ldots, o\mathrm{v}), \ldots, c_w(o_1, \ldots, o\mathrm{v})\}$.

4.1.1 Model Fragments and their Library

CM defines the model fragments as *abstract* descriptions of physical phenomena, containing only references to component types that can participate in these descriptions. Each model fragment relates a particular "view" of the structure and operation of a component inside a containing domain system, with a set of subcomponents, a set of structural connections between these sub-components and a set of qualitative descriptions of the phenomena taking place inside the component (involving reactions between the different participant subcomponents). In addition to the structural connections, a model fragment may also include a set of operating conditions that indicate whether its descriptions of phenomena hold when interacting with other fragments within a final model formulated from these fragments. Fragments can also contain a separate declaration of the quantities that are significant in describing the fragment-associated phenomena.

CM techniques therefore require a set of fragment definitions following these design principles. Individual fragments are then "instantiated" once the components, which participate in a given domain system environment, are determined. The basic definition of a model fragment forms the representational basis for the modelling of components and their accompanying phenomena as described in this thesis. This is formalised in the following definition.

Definition 4.1.1 (Model Fragment) *A model fragment μ is a tuple of $\langle P, Q, Cd^s, Cd^o, C\rangle$*

- $P(\mu) = \{p_1, \ldots, p_i\}$ *is a set of participant subcomponents.*
- $Q(\mu) = \{q_1, \ldots, q_j\}$ *is a set of quantities that help in the description of the fragment related physical phenomena. The quantities set can be further divided into the set of endogenous quantities (whose definition is provided by the fragment itself) and the set of exogenous quantities (independent with respect to any quantity defined by the model fragment).*
- $Cd(\mu) = Cd^s(\mu) \cup Cd^o(\mu)$ *is a set of preconditions, where* $Cd^s(\mu) = \{cd_1^s, \ldots, cd_k^s\}$ *is the set of structural conditions indicating how the represented component connects to other components via its terminal connections, and how its subcomponents are connected to each other, and* $Cd^o(\mu) = \{cd_1^o, \ldots, cd_l^o\}$ *is the set of operating conditions on the participants providing the basis for determining whether a component is "active" during the interaction with other components in a model of the domain system.*
- $C = \{c_1, \ldots, c_m\}$ *is a set of qualitative descriptions of the phenomena that represent the behaviour or function of a component. These descriptions are valid*

only when the model fragment conditions hold, i.e. when it is determined via the structural conditions that the component represented by the fragment is connected to other components and that it is active due to its operating conditions being fulfilled.

The notion of resolution of a fragment follows from the above definition:

Definition 4.1.2 (Resolution) *The resolution of a model fragment $R(\mu)$ is determined as the number of endogenous variables defined in a fragment.*

Resolution can regulate the amount of information contained in the model, and therefore can influence the complexity of the explanations communicated to the user for a particular component (Sime, 1998; Leitch et al., 1999). In this work the selection of fragments for each component is influenced by the determination of the desired fragment resolution. Identifying a proper resolution for particular model fragments is influenced by the system's perception of the user's expertise level.

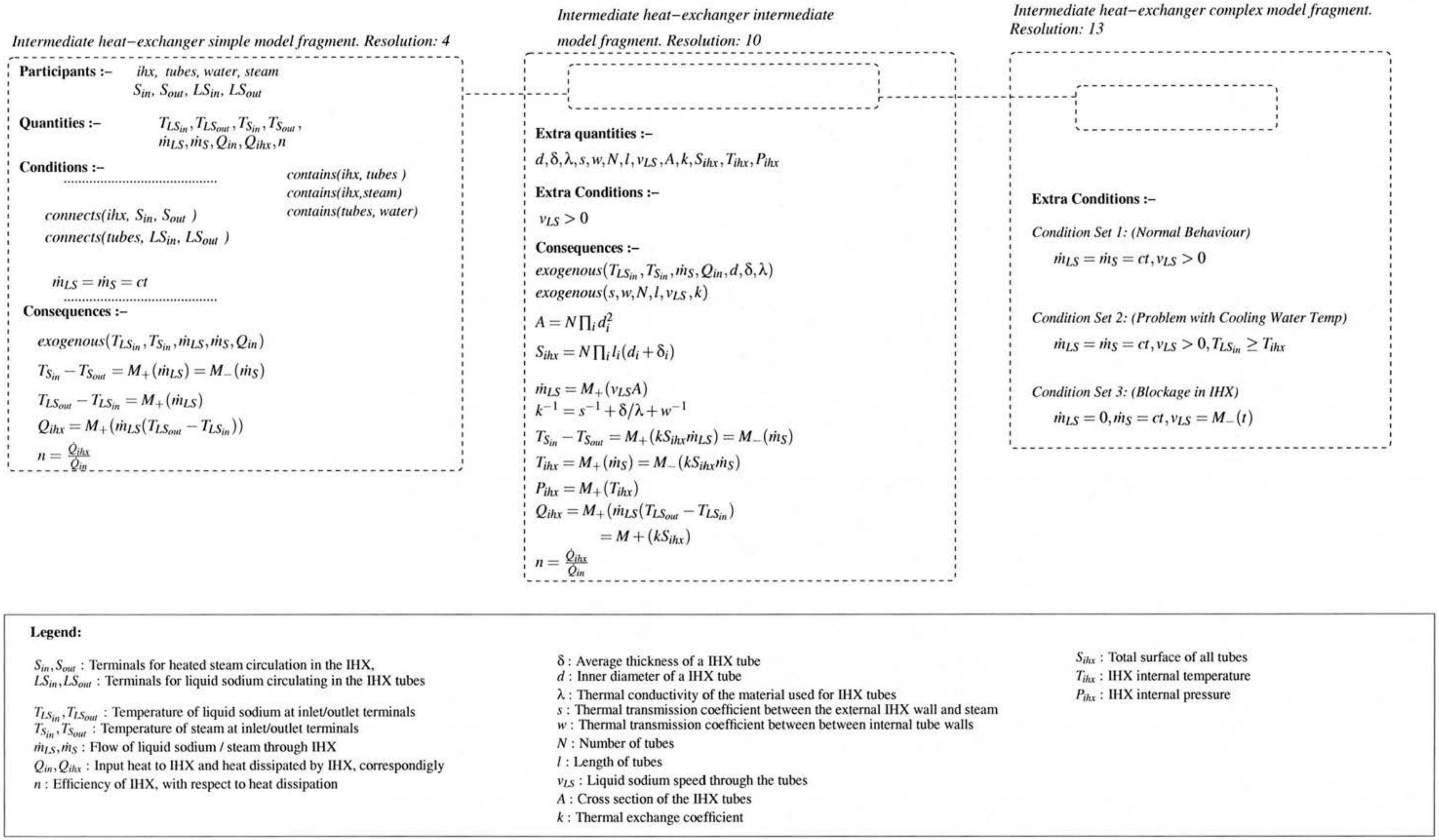
While the definition of the structure of a model fragment is fairly general, there are extra design constraints imposed over the type of information held by the model fragments for the particular purpose of explanation generation in this work. Specifically, each fragment may represent more than one of the structural, functional and behavioural properties of a particular component, depending on its resolution level. Fragments of the *lowest resolution*, which will hereafter be referred to as 'simple fragments', contain structural information for the modelled component (expressed via the set of structural conditions), the most significant variables involved and the typical qualitative constraints that describe the component's normal operation behaviour. Fragments of *intermediate resolution* encapsulate simple fragments and provide additional knowledge for the complete functional description of the component's normal behaviour. *High resolution* fragments cover the same information as the intermediate ones with additional alternative operating modes of the component: normal and faulty.

To illustrate these points concerning the differences of fragment resolutions, the description of the domain system in Section 3.4 is reused. Each fragment for the components of that domain can be envisaged as having two input and two output terminal connections, one for flow and another for pressure of the liquid sodium, allowing the interaction with the fragments of its adjacent components. A sample of model fragments for the component IHX is illustrated in Figure 4.1. The figure illustrates the structure of individual fragments, as well as the notion of encapsulating the information of fragments of lower resolution.

This work also uses an alternative interpretation of the grouping of fragments within *assumption classes*. Traditionally, model fragments for one single component are grouped as an "assumption class" for this component. CM requires that fragments within an assumption class should have different and perhaps even contradictory operating conditions. This arrangement is normally made in traditional CM to avoid inclusion of two alternative descriptions of the same phenomena in the same composable model. However, in this work it is the Bayesian network fragment selection mechanism that provides the means to automatically select one and only one fragment from the assumption class of each component, thus making this restriction, of demanding mutually contradictory conditions among fragments of a single assumption class, redundant. Note that the mechanism of how the Bayesian network is structured follows in Section 4.2.1.

Thus, from the viewpoint of the selection of fragments, the notion of an assumption class reduces in this work to being a group of fragments, each representing the same individual component. Different model fragments can be defined within one assumption class whose application conditions are no longer mutually exclusive. For instance, an overlapping of conditions is now allowed to occur between a high and an intermediate resolution fragments. High resolution, or complex, fragments are targeted especially at users who are perceived to be experts regarding the modelled phenomena, providing such users with an opportunity to identify alternative behaviour modes of a specific component.

Furthermore, the assumption classes are herein designed so that only one fragment per resolution level is allowed within the corresponding assumption class. If two or more fragments per resolution level co-exist in an assumption class, the selection algorithm would resolve any conflicts by randomly selecting a fragment out of the possible op-



Intermediate heat−exchanger simple model fragment. Resolution: 4

Participants :– *ihx, tubes, water, steam*
$S_{in}, S_{out}, LS_{in}, LS_{out}$

Quantities :– $T_{LS_{in}}, T_{LS_{out}}, T_{S_{in}}, T_{S_{out}},$
$\dot{m}_{LS}, \dot{m}_S, Q_{in}, Q_{ihx}, n$

Conditions :–

contains(ihx, tubes)
contains(ihx,steam)
contains(tubes, water)

connects(ihx, S_{in}, S_{out} *)*
connects(tubes, LS_{in}, LS_{out} *)*

$\dot{m}_{LS} = \dot{m}_S = ct$

Consequences :–

exogenous$(T_{LS_{in}}, T_{S_{in}}, \dot{m}_{LS}, \dot{m}_S, Q_{in})$

$T_{S_{in}} - T_{S_{out}} = M_+(\dot{m}_{LS}) = M_-(\dot{m}_S)$

$T_{LS_{out}} - T_{LS_{in}} = M_+(\dot{m}_{LS})$

$Q_{ihx} = M_+(\dot{m}_{LS}(T_{LS_{out}} - T_{LS_{in}}))$

$n = \frac{\dot{Q}_{ihx}}{\dot{Q}_{in}}$

Intermediate heat−exchanger intermediate model fragment. Resolution: 10

Extra quantities :–

$d, \delta, \lambda, s, w, N, l, v_{LS}, A, k, S_{ihx}, T_{ihx}, P_{ihx}$

Extra Conditions :–

$v_{LS} > 0$

Consequences :–

exogenous$(T_{LS_{in}}, T_{S_{in}}, \dot{m}_S, Q_{in}, d, \delta, \lambda)$
exogenous(s, w, N, l, v_{LS}, k)

$A = N\prod_i d_i^2$

$S_{ihx} = N\prod_i l_i(d_i + \delta_i)$

$\dot{m}_{LS} = M_+(v_{LS}A)$
$k^{-1} = s^{-1} + \delta/\lambda + w^{-1}$
$T_{S_{in}} - T_{S_{out}} = M_+(kS_{ihx}\dot{m}_{LS}) = M_-(\dot{m}_S)$
$T_{ihx} = M_+(\dot{m}_S) = M_-(kS_{ihx}\dot{m}_S)$
$P_{ihx} = M_+(T_{ihx})$
$Q_{ihx} = M_+(\dot{m}_{LS}(T_{LS_{out}} - T_{LS_{in}})$
$= M + (kS_{ihx})$
$n = \frac{\dot{Q}_{ihx}}{\dot{Q}_{in}}$

Intermediate heat−exchanger complex model fragment. Resolution: 13

Extra Conditions :–

Condition Set 1: (Normal Behaviour)

$\dot{m}_{LS} = \dot{m}_S = ct, v_{LS} > 0$

Condition Set 2: (Problem with Cooling Water Temp)

$\dot{m}_{LS} = \dot{m}_S = ct, v_{LS} > 0, T_{LS_{in}} \geq T_{ihx}$

Condition Set 3: (Blockage in IHX)

$\dot{m}_{LS} = 0, \dot{m}_S = ct, v_{LS} = M_-(t)$

Legend:

S_{in}, S_{out} : Terminals for heated steam circulation in the IHX,
LS_{in}, LS_{out} : Terminals for liquid sodium circulating in the IHX tubes

$T_{LS_{in}}, T_{LS_{out}}$: Temperature of liquid sodium at inlet/outlet terminals
$T_{S_{in}}, T_{S_{out}}$: Temperature of steam at inlet/outlet terminals
$\dot{m}_{LS}, \dot{m}_S$: Flow of liquid sodium / steam through IHX
Q_{in}, Q_{ihx} : Input heat to IHX and heat dissipated by IHX, correspondigly
n : Efficiency of IHX, with respect to heat dissipation

δ : Average thickness of a IHX tube
d : Inner diameter of a IHX tube
λ : Thermal conductivity of the material used for IHX tubes
s : Thermal transmission coefficient between the external IHX wall and steam
w : Thermal transmission coefficient between between internal tube walls
N : Number of tubes
l : Length of tubes
v_{LS} : Liquid sodium speed through the tubes
A : Cross section of the IHX tubes
k : Thermal exchange coefficient

S_{ihx} : Total surface of all tubes
T_{ihx} : IHX internal temperature
P_{ihx} : IHX internal pressure

Figure 4.1: Model fragments excerpt for the component Intermediate Heat Exchanger (IHX).

tions. If there are conflicts that reasoning at the resolution dimension alone cannot handle then this is indicative that there is a need for another modelling dimension, e.g. abstraction, scope, accuracy (Leitch et al., 1999), to be employed to help make further parameterisation of the fragment. The implementation of this idea is, however, beyond the scope of the present investigation as it does not affect the actual approach proposed when addressing the resolution dimension only.

The set of assumption classes (i.e. the set of fragments for each component) is then collectively identified as part of the "library" of model fragments. There can be only one assumption class per component. A library is parameterised to distinguish sections in it that contain assumption classes for components of different physical domains. The following definition provides some notational basis for these aspects of the library:

Definition 4.1.3 (Model Fragment Library) *A model fragment library L is a tuple of $\langle A, D \rangle$ where:*

- $A(C) = \{\alpha_1, \ldots, \alpha_N\}$ *is a set of pre-specified N assumption classes.*
- $D = \{d_1, \ldots, d_k\}$ *is a set of domain system definitions, each including a set of references to those assumption classes that contain representations for the components which participate in the corresponding domain system. The following constraint is imposed: for each component that appears in one domain system definition there can be only one assumption class representing the component.*

Allowing just one assumption class per component per domain in the library of model fragments allows the compositional inference to focus on the phenomena of the particular physical domain that the class is defined under. This follows the basic presumption which holds for all phenomena used as examples: they originate from a single physical domain. If there is a need to explain phenomena from two or more domains for isolated components, then a methodology for mixing assumption classes is necessary. To cope with this requirement it is not sufficient to use resolution alone for the structuring of the Bayesian network and for the selection algorithm, as argued previously. This would allow combination of the separate assumption classes to allow interaction of the fragment selection procedures.

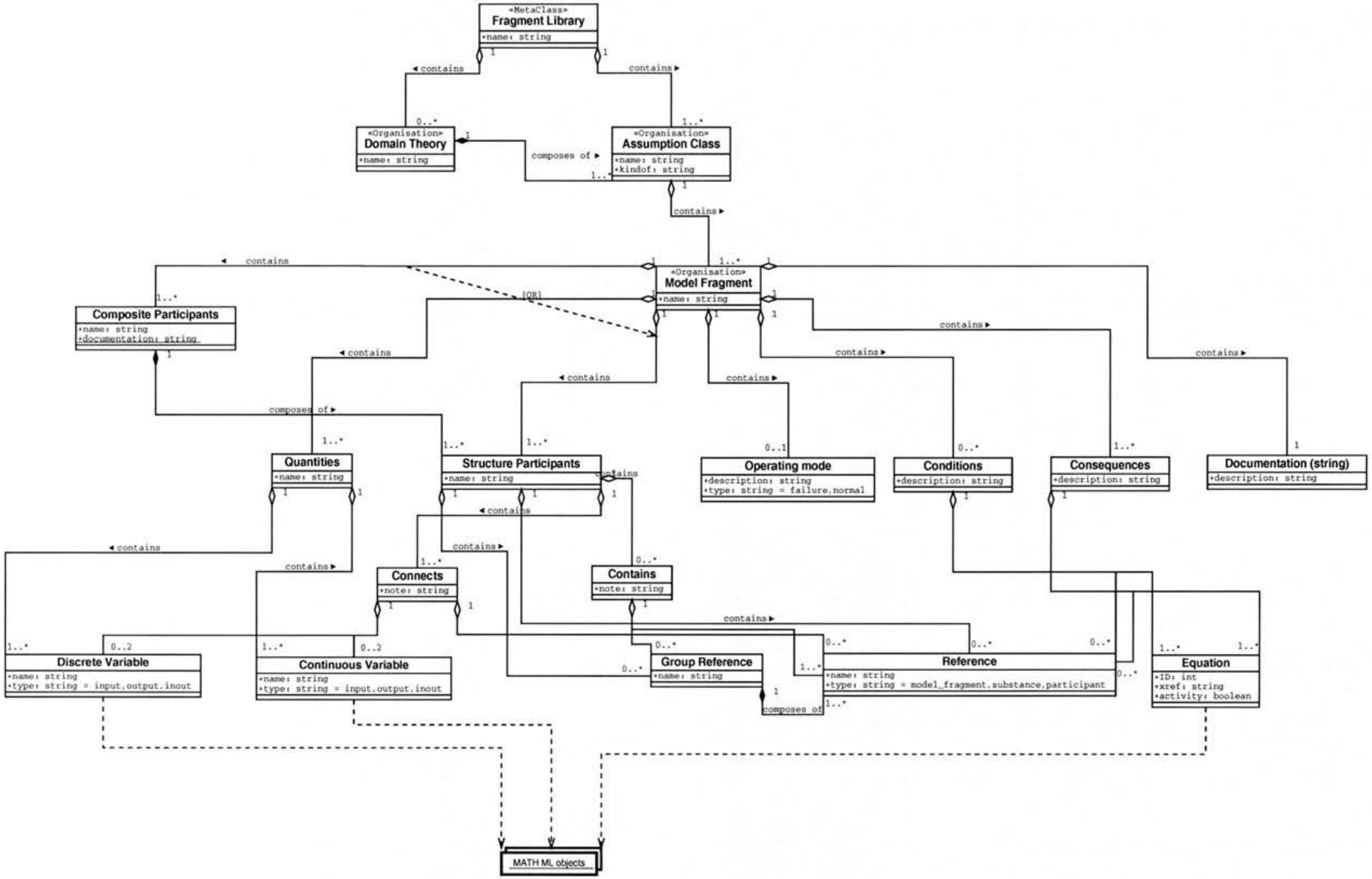


Figure 4.2: XML DTD associations between domain representation constructs.

The software implementation of the representation of fragments, assumption classes and domain library is facilitated by the use of XML. The representation uses an application-specific XML document type definition (DTD) (Ray, 2001), which enables definition of all the domain representation elements in a concise and unambiguous manner. In addition to usual usefulness in reducing coding errors and code complexity, the use of XML as representation apparatus eases the passage of information to other modules in the framework, such as the explanation generation Content Handler that determines the content to be presented back to the user. In general the grammar for the representation of library, assumption classes and model fragments is illustrated in Figure 4.2.

4.1.2 Scenario Description

A scenario in this work amounts to a structural description of the domain system for which explanations are required. This involves specifying the components that are involved and the interconnections between these components, whilst also indicating the type of phenomena being modelled, through the interconnections between components. In this aspect, the scenario is an abstraction of a model for the domain system that is being represented. It is an accurate representation of the real-world system only with respect to the interconnections of the various components. The scenario is used once at the initialisation stage of the model formulation, assisting in the construction of the Bayesian network that is used for the model fragment selection.

An example of a scenario for the example domain system in Section 3.4 is given, in predicate format, in Figure 4.3.

This is a basic scenario that identifies all possible components for the device given in Section 3.4. All the component predicates are named based on the names of the corresponding assumption classes that contain fragments representing these components. This name matching is crucial for the determination of model fragments that can represent these components.

Given the above scenario description, the Model Formulator can select assumption classes that represent the components indicated in the scenario, whilst considering only sodium_source(SE)

hydraulic_connection(SE, X_1, X_6)

hydraulic_resistance(R_1)

hydraulic_connection(R_1, X_1, X_2)

intermediate_heat_exchanger(IHX)

hydraulic_connection(IHX, X_2, X_3)

hydraulic_resistance(R_2)

hydraulic_connection(R_2, X_3, X_4)

evaporator(EV)

hydraulic_connection(EV, X_4, X_5)

hydraulic_resistance(R_3)

hydraulic_connection(R_3, X_5, X_6)

Figure 4.3: Scenario for the example system in Section 3.4.

assumption classes that represent hydraulic phenomena. The choice of this phenomena to cover is determined by the presence of a classifier for the type of the connections between fragments. In the example the classifier "hydraulic" determines that both components should come from an assumption class within the domain of hydraulic phenomena in the library of model fragments. The matching of the assumption classes to the phenomena of interest indicated in the scenario is done via simple string matching based on the assumption class name and the phenomena identifier in the scenario connection predicates.

Finally, each connection statement in the scenario description has to determine the component for which the connection is described, and the pair of terminals between which the connection spans. The order of the terminals should always be input first, output last. If a particular component has multiple inputs and/or multiple outputs then separate connection statements are required to cover all connectivity combinations between the inputs and outputs.

The applicability of a model fragment with respect to a specified scenario follows the definition below:

Definition 4.1.4 (Applicability of a Model Fragment) *A model fragment μ is applicable with respect to a scenario $S = \langle O, C \rangle$, if*

1. *The scenario description contains a statement, within the set of O object statements, describing the presence of a component with the same name as the assumption class to which the said fragment belongs.*
2. *The scenario statements for the connection of a component, which are part of the C set of statements, indicate that the types of phenomena sought after for representation in the final model are the types of phenomena represented by the μ's assumption class.*
3. *All the structural conditions in the fragment are entailed in the scenario description. That is, the scenario statements that determine the connectivity of a component to its terminal inputs and outputs are also part of the said fragment's connectivity descriptions.*

For the checks for the structural conditions of a fragment to comply with the structural connectivity requirements present in the scenario, it suffices to ensure that the candidate fragments contain a 'structural-participants' XML section with a 'connects' sub-section indicating connectivity to the same number of terminals as indicated in the scenario. Again, this can be done via string matching.

4.2 Creation of the Bayesian Network

Based on the inputs as described in the previous Section and the structure of a Bayesian network the Model Formulator can proceed in determining which model fragments are most appropriate for the given problem. The structuring of the Bayesian network is described next, covering both the qualitative aspects (arrangement and interconnections between the network nodes) and the quantitative ones (assignment of probabilities to the network).

4.2.1 Bayesian Network Structure

The scenario showing the structural connections of the domain system components is used to build a Bayesian network relating model fragments from one assumption class to fragments of other classes. Through simple string pattern matching, the connection relationships of the scenario are analysed to determine the assumption classes that can be used to represent the individual components. Each fragment from an assumption class is considered to be a candidate for the representation of the corresponding component.

Each node of the network stands for the selection or rejection of a specific model fragment. As such, a node may take a value from the set $\{yes, no\}$ (with *yes* indicating the selection of the fragment and *no* the rejection). The links of the Bayesian network represent the relation between model fragments as determined by the structural description of the domain system: If component A has its output connected to the input of component B, then links are established from the nodes which represent model fragments of A toward nodes denoting component B.

Generally speaking, in CM one should be careful when "connecting" fragments through terminal connections that can stand for different accuracy levels for the underlying quantities. However, the model fragments used by the Formulator are of qualitative symbolic nature and such connections can be reasoned on the grounds of mapping qualitative values between the variables involved in the fragments. In absence of any prohibiting connection constraints in the scenario, the Bayesian Network construction mechanism assumes the worst case, allowing all model fragments of one component A to be connected to all fragments of another component B, provided that A is physically connected to B. Any given constraints from theoretical or empirical knowledge sources that prohibit such a connection can be expressed in the scenario description and can be taken into account in setting up the network structure, by assigning a prior probability of zero to the forbidden network links.

For training on the operation of a given physical system, as it is the focus in the present work, sufficient knowledge of the structure of the system concerned is known. This eases the task of devising the structure of the Bayesian network required. In most cases, such a task becomes straightforward. Indeed the structure is directly derivable from the interconnections of components in the scenario description, layering the nodes for each assumption class and connecting them in a top-down fashion starting with the component that is the source of the flow of energy in the entire system. This structure, although unorthodox when viewed with respect to the traditional approaches in Bayesian networks, offers the necessary interconnections between the network nodes that drive the decisions for fragment selection when evidence arrives. The decision of how network nodes for different model fragments can be interconnected, for the purposes of decision making, needs also to be conditioned on the required d-separation relationships (Pearl, 2000) within the network.

As described in (Pearl, 2000), d-separation can be identified as follows: in causal chains $i \rightarrow m \rightarrow j$ and causal forks $i \leftarrow m \rightarrow j$, the two extreme variables are marginally dependent but become independent of each other once conditions are imposed over the middle variable (e.g. its value is known). Thus, if the value of the middle variable is given, then learning for the value of i has no effect on the probability of j. On the other hand, inverted forks $i \rightarrow m \leftarrow j$, representing two causes having a common effect acts in the opposite way: If the two extreme variables are (marginally) independent, they will become dependent once conditions are imposed on the middle variable or any of its descendants. Thus, if a certain piece of evidence is subsequently received or, say, any increase of the probability for i is established, then the probability for j will automatically be decreased.

The d-separation is important for the case of selecting fragments for model composition. This can be exemplified better following the illustration of Figure 4.8. Suppose that evidence is provided for one of the components in assumption class R_1. Because the nodes for the R_1 fragments d-separate the nodes in assumption classes for PM and IHX, the nodes for the two latter components are rendered independent from each other. This means that selection of component fragments for IHX is not relevant in any way selection of fragments for PM. However, knowledge of which fragment is selected for component R_1 renders the nodes for component PM dependent on each other. Therefore, if subsequent evidence comes for any of the three nodes in PM then the probabilities for the other two nodes will have to be adjusted accordingly. Interestingly, d-separation also accommodates that knowledge for the selection of any of the IHX fragments will also render the probabilities of selecting any of the PM components dependent upon each other, conditioned on the selection of IHX fragments.

4.2.2 Assignment of Prior Probabilities

Given the structure of a Bayesian network, each network node is annotated with estimates for the conditional probabilities of the node acquiring a *yes* or *no* value when its "parent" nodes are assigned their values. In general, deriving such prior probabilities is a difficult task (Pearl, 1988). There are applications where historical data is available, thereby enabling the required estimation, sometimes by simply calculating the frequencies of value appearance. In less fortunate cases, like the present work where such frequencies are unavailable, rules or heuristics have to be derived from the problem description in order to decide the prior probabilities. If however, enough prior expertise exists for the connection between the available fragments then the probability measures can be adjusted to take this knowledge into account.

The heuristics employed herein depend on whether or not a node is a root node, i.e. one that has no parent nodes. The first heuristic indicates whether a model fragment is to be selected, by weighing the following two important factors:

- The strength of causal influence that the fragment receives from its selected parents, and
- The compliance of the fragment with respect to the resolution level of these parent nodes.

Intuitively, the more causal influence a fragment receives from its selected parents, and the closer its resolution to the parents' resolution level, the more chance there is for the fragment to be selected. This heuristic can be stated as follows :

Definition 4.2.1 (Prior Probabilities for the Non-Root Nodes) *For each non-root node MF_{ji}, $j = 1, \ldots, L$, representing the selection (yes) or rejection (no) of model fragment j of component C_i, $i = 1, \ldots, M$, with parent nodes $U_{jk}, k = 1, \ldots, N$, the prior probability of selecting the corresponding fragment when certain parent nodes take the value yes and the other parent nodes take the value no, is determined by:*

$$P(MF_{ji} = yes | \bigcap_p U_{jp} = yes) \quad = \quad \frac{Opt_{ji} \cdot R_{ji(\text{influenced})} \cdot \prod_r V_{MF_{ji} \leftarrow U_{jr}}}{1 + \prod_r (R_{MF_{ji}} - R_{U_{jr}})^2}$$

In this heuristic p ranges over the parents that have taken the value *yes*. The denominator of the above formula denotes a combination of the quadratic difference of the resolution level between a fragment and each of its selected parent fragments. Opt_{j_i} stands for the probability that the particular fragment MF_{j_i} will be selected assuming that the resolutions of all fragments are arranged based on a probability distribution, such as a Gaussian (Moore and McCabe, 2002). $R_{j_i(influenced)}$ denotes the ratio of fragment variables which are influenced by variables of the particular parent fragments combination that are assumed to be selected each time: $R_{j_i(influenced)} = N_{j_i(influenced)}/N_{j_i(total)}$, where $N_{j_i(influenced)}$ is the total number of variables of MF_{j_i} being influenced directly by the union of influencing parent fragments considered, and $N_{j_i(total)}$ is the total number of variables of MF_{j_i}. $V_{MF_{j_i} \leftarrow U_{jl}}, l \in \{p, r\}$ denotes an estimate of the amount of causal influence that fragment MF_{j_i} receives from parent U_{jl} when U_{jl} is *yes*. Let $N_{influence}$ be the number of those variables which are defined in the consequences of U_{jl} and which influence the variables involved in MF_{j_i}, and N_{total} be the total number of variables defined in U_{jl}, then $V_{MF_{j_i} \leftarrow U_{jl}}$ is calculated as the ratio $N_{influence}/N_{total}$.

As an example of the heuristic, consider the simple network of Figure 4.4 which connects four model fragments of different detail levels (for three components A, B and C). Given the resolutions for each of the fragments, a probability distribution of the likelihood of a fragment being selected based only on its resolution can be calculated, and in the case of fragment B, this is $\text{Opt}_{B_1} = 0.306588$. The ratio of the variables of fragment B_1 which are influenced by variables of either A_1 or A_2 is $R_{B_1(influenced)} = 1/4 = 0.25$. Furthermore, model fragment $A1$ has one variable out of a total two variables that can influence variables in its child node, fragment $B1$, that is $V_{B_1 \leftarrow A_1} = 0.5$. Similarly $V_{B1 \leftarrow A_2} = 1/3 = 0.33$. The difference in detail between fragments B_1 and A_1 is $D_{B_1} - D_{A_1} = 4 - 2 = 2$. The application of the above heuristic for B_1 gives, thus, the following non-normalised results:

$$
\begin{aligned}
P(B1 = yes|A1 = yes, A2 = yes) &= \frac{0.306588 \cdot 0.25 \cdot 0.5 \cdot 0.33}{1 + ((4-2)^2 \cdot (4-3)^2)} = 0.00252935 \\
P(B1 = yes|A1 = yes) &= \frac{0.306588 \cdot 0.25 \cdot 0.5}{1 + ((4-2)^2)} = 0.0076647 \\
P(B1 = yes|A2 = yes) &= \frac{0.306588 \cdot 0.25 \cdot 0.33}{1 + ((4-3)^2)} = 0.0126468
\end{aligned}
$$

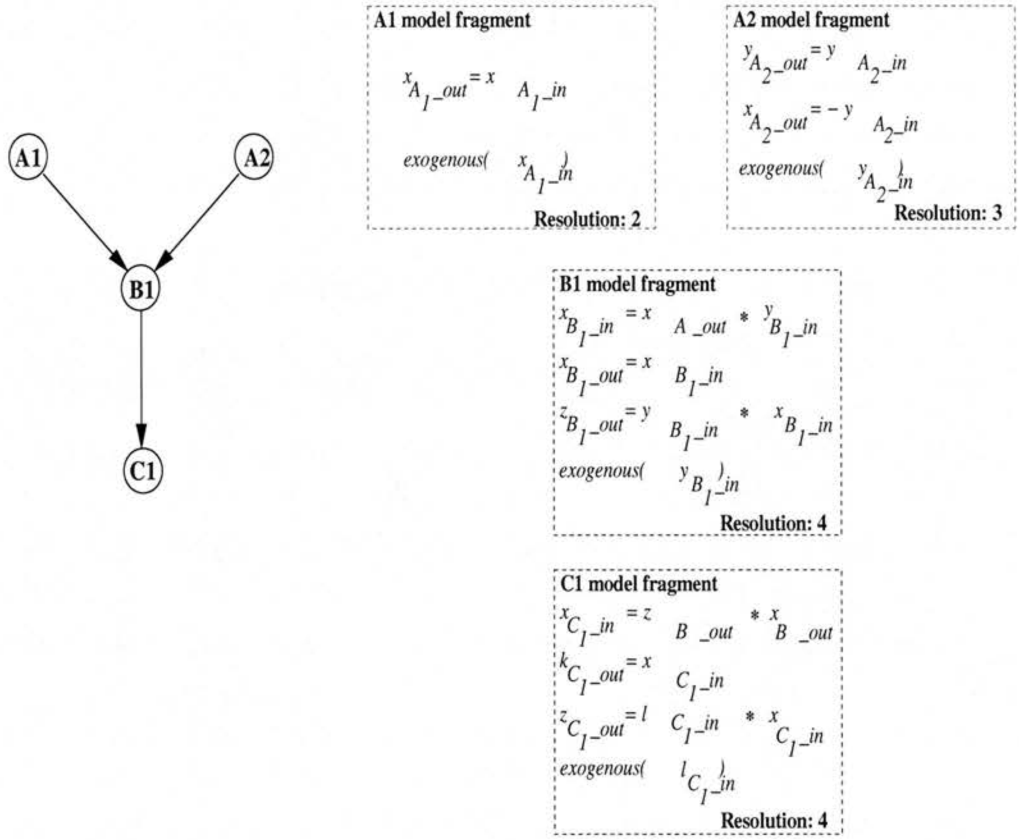

A1
A2
B1
C1
A1 model fragment
$x_{A_1\_out} = x_{A_1\_in}$
$exogenous(x_{A_1\_in})$
Resolution: 2
A2 model fragment
$y_{A_2\_out} = y_{A_2\_in}$
$x_{A_2\_out} = -y_{A_2\_in}$
$exogenous(y_{A_2\_in})$
Resolution: 3
B1 model fragment
$x_{B_1\_in} = x_{A\_out} * y_{B_1\_in}$
$x_{B_1\_out} = x_{B_1\_in}$
$z_{B_1\_out} = y_{B_1\_in} * x_{B_1\_in}$
$exogenous(y_{B_1\_in})$
Resolution: 4
C1 model fragment
$x_{C_1\_in} = z_{B\_out} * x_{B\_out}$
$k_{C_1\_out} = x_{C_1\_in}$
$z_{C_1\_out} = l_{C_1\_in} * x_{C_1\_in}$
$exogenous(l_{C_1\_in})$
Resolution: 4

Figure 4.4: A simple Bayesian network and the corresponding model fragments.

A much simpler heuristic is used to assign prior probabilities to root nodes, assuming all fragments of a component are initially equally likely to be selected:

Definition 4.2.2 (Prior Probabilities for the Root Nodes) *For all root nodes RMF_{r_i}, $r = 1,\ldots,R$ of a component C_i, $i = 1,\ldots,S$, the prior probability of them being selected is $P(RMF_{r_i} = yes) = \frac{1}{R}$.*

The presumption about equal likelihood of selection for the root node fragments is based on the fact that these nodes have no connections to parent fragments, denoting absence of possible direct influences in the decision of whether or not to select them for the final model. Indeed, without prior knowledge on the possible ways that the fragments can interrelate to each other, the only assumption that one can make over how to select any of the available fragments is to choose randomly one from each assumption class. In the case of the non-root nodes this is the starting point for the calculation of their prior selection probability. However, due to the interconnections between fragments of different components the final probability for non-root nodes accommodates terms that provide the effects of this connectivity when propagating evidence. The influence on the selection of root node fragments based on the provided evidence eventually is accounted for by the inference algorithm employed, via the aggregation of evidence and priors while propagating the evidence around the network.

The heuristics above provide a mechanism to compute the prior probabilities for the selection of the fragments associated with the network nodes, i.e. they provide the prior probabilities for the nodes to take a value *yes* with various combinations of the values for their parent nodes. The prior probabilities for the nodes to take the *no* value can be calculated using the principle of *maximum entropy* (Pearl, 1988), subject to the constraint that the sum of all prior probability values to a node must be equal to 1 and assuming a uniform distribution of prior probabilities for which no previous estimation is available. Assuming that there are n probability values in total then the entropy is defined as:

$$H(p_1,\ldots,p_n) = -\sum_{i}^{n} p_i \cdot ln(p_i)$$

Attempting to maximise H over p_i, when m of the n probability values are already known, with $m \neq n$, leads to the remaining probability values being determined by:

$$p_j = \exp^{-(\beta+1)}$$

where $\beta = ln(\frac{m-n}{(-1+\sum_i^m p_i)\cdot e})$, summing over all the known probabilities for the node that is considered. The value of β can be found by following the solution below:

$$\begin{aligned} \sum_{j=m+1}^{n} p_j + \sum_i^m p_i &= 1 \Rightarrow \\ \exp^{-(\beta+1)} \cdot (n-m) + \sum_i^m p_i &= 1 \Rightarrow \\ \exp^{(\beta+1)} &= \frac{m-n}{-1+\sum_i^m p_i} \Rightarrow \\ \beta+1 &= ln(\frac{m-n}{-1+\sum_i^m p_i}) \Rightarrow \\ \beta &= ln(\frac{m-n}{(-1+\sum_i^m p_i)\cdot e}) \end{aligned} \tag{4.1}$$

4.3 Model Formulation

With the Bayesian network structure and priors defined, reasoning is performed based on the evidence of which fragments of what components are of current interest to the user and on which resolution level is indicated for certain fragments by the Approximate User Monitor. The evidence is translated to the selection of the indicated fragments with a probability value of 1. This is propagated throughout the network to determine the posterior probabilities of the nodes for which there exists no evidence, using the standard Bayesian network inference method as detailed in Section 2.4.3.2. The method is hereafter referred to as the *fragment selection algorithm*.

The algorithm relies on a probability-based valuation system (Shenoy and Shafer, 1988), but before delving into the theoretical details it would be beneficial to establish the language used for its presentation below. Thus, consider a model formulation Bayesian network. In reality, its nodes represent variables. Each variable may range over a finite set of possible values, known as the *frame* of the variable, e.g. the values

$\{\text{yes}, \text{no}\}$ assigned to each of the network nodes in the model formulation Bayesian network. A *configuration* of a finite non-empty set of variables is an element of the Cartesian product of the frames of the variables in this set. In the following descriptions, X denotation is used for a set of variables ; g, h, q for subsets of X ; W_g for the set of configurations of g ; $x_{\text{single}}, y_{\text{single}}$ for single configurations ; $a_{\text{set}}, b_{\text{set}}$ for sets of configurations. Finally, configurations of one set of variables need sometimes to be extended or projected to another set. Thus a configuration x_{single} of set g is extended to set h, where $g \subset h$, by building the Cartesian product between x_{single} and W_{h-g}. On the other hand, projecting the configuration x_{single} of set g to set k, where $k \subset g$, is performed by removing elements in x_{single} that belong to $g - k$. Extension is denoted by $x_{\text{single}}^{\Uparrow h}$, whereas projection is denoted by $x_{\text{single}}^{\Downarrow k}$.

Having established the notation and keywords that are used in the discussion that follows, the concept of valuations can now be presented. Intuitively, valuation systems associate sets of variables with *valuations*, objects encoding information about the relationships between these variables, for the various configurations of their values, and use two operators for combining and transferring valuations. Typically, valuations can be determined as probability or possibility distributions, boolean values or any other quantitative representation used in uncertainty calculus, whilst the corresponding operators can be determined based on the type of uncertainty representation chosen (Saffiotti and Umkehrer, 1991). Thus, valuation-based systems can in fact be deemed as providing an abstraction, or a "language" of describing uncertainty in quantitative terms.

In the case of a model formulation network, a valuation system is formed on the basis of the set of variables, mapped $1-1$ to the set of the network nodes, and representing the selection of model fragments in conjunction with their value sets ($\{yes, no\}$) together with the prior probabilities associated with each node taking either of these values whilst considering all possible combinations of parent node values. The set of probabilities and values associated with each network node is the node valuation. Two operators are defined to act on node valuations upon propagating evidence:

- The combination operator used for aggregating the different valuations of the network nodes. Intuitively, combination corresponds to aggregation of *knowledge*. If V_R and V_S are valuations for the sets of variables R and S, representing knowledge about variables in variable sets R and S, then the combination $V_R \otimes V_S$ represents the aggregated knowledge about variables in $R \cup S$.

- The marginalisation operator narrows the focus of interest for a verdict on the probabilities of a subset of variables. Intuitively, the marginalisation operator, denoted with $\downarrow$, corresponds to coarsening of knowledge. If V_S is a valuation for a set of variables variable set S, i.e. representing some knowledge about variables in S, and $t \in S$ is a variable from S, then the marginalisation denoted $V_S^{\downarrow(S-\{t\})}$ represents the knowledge about variables in $S - \{t\}$, as implied by V_S if we disregard variable t. Thus given a combined valuation on a set of variables the marginalisation operator allows the analysis of this joint valuation to the valuations of its part variables.

The evaluation process, i.e. propagation of evidence around the network, amounts to computing the global valuation for the network by combining all the valuations of the network nodes and finding the marginals of the global valuation for each node that has been given no evidence from the Approximate User Monitor. Calculating explicitly the global valuation is often infeasible from the computational viewpoint. However, as described in (Shenoy and Shafer, 1988), there exists a local computation schema for evaluating valuation systems. It is local in the sense that combinations of valuations can be performed without extending each node valuation to the whole space of possible valuations for the entire network. This schema can be applied for the propagation of probabilities in the model formulation network, if the combination and marginalisation operators employed satisfy the following three axioms:

The combination operator is commutative and associative This allows the combination of valuations in any order. Thus, it would not matter the direction in which evidence is propagated toward, in the model formulation Bayesian network. Therefore, instead of combining all the network valuations and then applying the evidence to the combined valuation, it suffices to combine the evidence locally with the valuations of the nodes involved, and then extend the combination to the neighbouring nodes and so forth till the entire network is updated.

The marginalisation operator is consonant Regarding marginalisation as a coarsening of a valuation by deleting variables, then this axiom determines that the order in which variables are deleted does not matter. Thus after evidence is propagated within a network, and marginalisation is applied to discover the resulting valuation of some of the nodes, the order in which nodes are removed during marginalisation does not matter. This enables performing local marginalisation operations as evidence gets propagated instead of performing it after the entire set of valuations for the Bayesian network is updated with the incoming evidence. From an efficiency and accuracy point of view this improves valuation computation.

The marginalisation operator is distributive over the aggregation operator Whereas the previous two axioms set the underground for local computation, this axiom finally makes local computation possible. It simply states that computing the aggregate of two valuations V_R and V_S and marginalising over a variable $t \in S$ need not be done in the order $(V_R \otimes V_S)^{\downarrow((R \cup S)-\{t\})}$, but it can be done instead as $V_R \otimes (V_S^{\downarrow(S-\{t\})})$. Thus when propagating evidence, it is not necessary to combine valuations first and marginalise afterwards to find the effect of the evidence, but instead local marginalisation can take place first to produce valuations focused on the variables of interest and combination to valuations already updated with incoming evidence can happen using the marginalised focused valuations. Efficiency-wise this is also important since marginalisation over a smaller set of values (as in the case of $S - \{t\}$ instead of $(R \cup S) - \{t\}$) is more efficient.

Using probabilities as the entities to represent valuations in the model formulation network allows employment of the traditional probability operators to stand in for the combination and marginalisation of distributions. Formally:

Valuations The valuation of a node in the model formulation Bayesian network is the probability distribution of the node for all configurations between the values for the variable which it represents and the values of all its parent nodes.

Combination If Γ and H are probability distributions on node sets g and h, then their combination is the joint probability distribution on $g \cup h$ defined as:
$(\Gamma \otimes H)(x_{\text{single}}) = \Gamma(x_{\text{single}}^{\Downarrow g}) \cdot H(x_{\text{single}}^{\Downarrow h}) \quad \forall x_{\text{single}} \in W_{g \cup h}$.

Marginalisation If $h \subseteq g$ and Γ is a probability distribution on g, then the marginal of Γ on h is the (conditional) probability distribution on h defined by:
$\Gamma^{\downarrow h}(x_{\text{single}}) = \sum\{\Gamma(x_{\text{single}}, y_{\text{single}}) | y_{\text{single}} \in W_{g-h}\} \quad \forall x_{\text{single}} \in W_h$
whilst assuming that $\Gamma^{\downarrow h}(x_{\text{single}}) = \Gamma(x_{\text{single}})$ if $g = h$.

Furthermore, relating to the aforementioned axioms, the product operator is both commutative and associative, the summation operator is consonant (the order of variables over which sums are performed does not matter to the end result) and it is also distributive over the product operator. These properties guarantee that these operators will support local computations as required by the Shenoy-Shafer valuation-based inference.

Given the combination and marginalisation operators, the reasoning process of the Bayesian network follows exactly the same process outlined in Section 2.4.3.2. To facilitate the description herein a summary of the reasoning steps followed is given below, exemplified with respect to the simple Bayesian network in Figure 4.5:

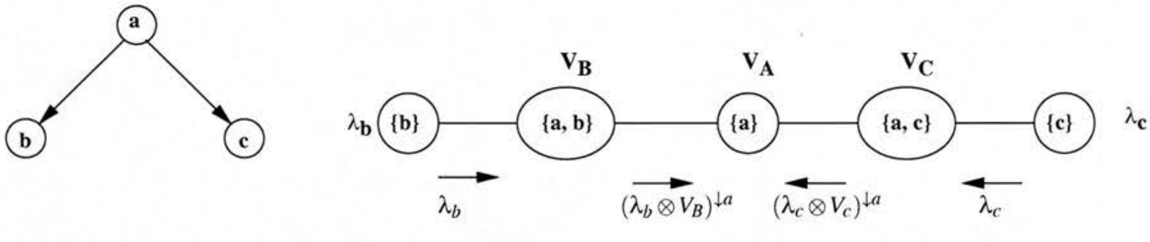


Figure 4.5: A simple Bayesian network and the corresponding join tree, with messages used for the calculation of the posterior distribution for node a.

The depicted network includes probability distributions – or valuations: V_A for set $\{a\}$ representing the prior probability distribution for a, V_B for $\{a,b\}$ representing the conditional probability distribution for b given a, and V_C for $\{a,c\}$ representing the conditional probability distribution for c given a. Evidence for the leaf nodes b and c are denoted as λ_b and λ_c respectively.

The Shenoy-Shafer inference works in three phases as outlined below:

Join tree construction phase During this phase, first a *hypergraph* of the domains of the valuations is constructed. In general hypergraphs are a generalisation of common graphs, where the number of nodes connected by an edge is not limited to two. Their so-called *hyperedges* may connect to an arbitrary number of nodes. For the Bayesian network in Figure 4.5 the hypergraph is $H = \{\{a\},\{a,b\},\{a,c\},\{b\},\{c\}\}$ as a list of hyperedges connecting subsets of the set of the network nodes appearing in the depicted network. Subsequently, the hyperedges of H are rearranged in a binary join tree, as the one indicated next to the Bayesian network in Figure 4.5. Finally each valuation (including the available evidence) is associated with the corresponding join-tree node, as indicated in the same figure.

Message passing phase The target is to find the joint valuation (e.g. joint probability distribution) for those subsets of variables which are of interest. The nodes for which marginals are desired request messages from their neighbours. These messages effectively carry out the combination of external evidence with the distributions associated with the join tree nodes (using the aggregation and marginalisation operators). Each node adds its own influence to the message that is received from its neighbours prior to sending it further in the join tree. Message passing seizes when there is no more change in the valuation of each join tree node due to the messages arriving from its neighbours. To calculate the posterior distribution for a variable the messages indicated in Figure 4.5 are passed from the evidence nodes b, c.

Marginalisation phase Finally, in this phase of the reasoning, the message passing has stopped and the marginals for the desired variables are computed, using the marginalisation operator (summation in this application) as defined in paragraphs above. For the calculation of the marginal on variable a, i.e. the posterior probability distribution for that variable, the messages collected from the neighbouring nodes are combined with the valuation V_A for a giving the result for the posterior a's distribution, V'_A: $V'_A = V_A \otimes (\lambda_b \otimes V_B)^{\downarrow a} \otimes (\lambda_c \otimes V_C)^{\downarrow a}$.

In implementation, the Model Formulator as a mechanism is based on the system `PULCinella` (Saffiotti and Umkehrer, 1991). `PULCinella` is a general tool for propagating uncertainty through the local computation technique of Shafer and Shenoy as outlined above in this section and detailed in (Shenoy and Shafer, 1988). Algorithm 4.3.1 describes the top level propagation algorithm in `PULCinella`.

ALGORITHM 4.3.1: PROPAGATE_VALUATION_SYSTEM(*nodes*,*relations*)

```
if $relations \neq \emptyset$ then
    comment: Pulcinella uses Markov trees to represent join trees
    comment: Generate the Markov tree to start with, if not generated
    if not MarkovTree?($nodes \cup relations$)
      then $mtree \leftarrow$ GENERATE_MARKOV_TREE($relations$)
    $mvertices \leftarrow$ VERTICES($mtree$)
    PROPAGATE_UP(EVIDENCE_NODES($mvertices$))
    PROPAGATE_DOWN()
    NORMALISE_DISTRIBUTIONS($mtree$)
```

The algorithm follows the theory in the implementation of the required phases for the completion of the network representation used and the inference procedure. First the join tree is constructed, as a Markov tree, following these steps:

1. Represent the valuation system by a hypergraph H, similar to the example provided above in relation to the Figure 4.5. Each node of the original Bayesian network is associated to a hyperedge. The probability distribution for each of the Bayesian network nodes appearing in a join tree node is associated to its corresponding hyperedges. The hypergraph is *reduced* so that:

    - Every hyperedge containing at least one unique node, i.e. not contained in any other hyperedge, is removed. This causes the momentary removal of the leaf nodes of the original Bayesian network (i.e. all nodes without children), in order to work first on making the right combinations of intermediate nodes in the network.

- Every hyperedge $E_i \in H$ is *minimal*, i.e. there is no other hyperedge $E_j \in H,\ i \neq j$ such that $E_j \subseteq E_i$. This guarantees that there are no redundancies in the hyperedges included in H.

In the reduction process, the singular node sets, each of which contains only one node from the Bayesian network, are also removed to facilitate the processing of the sets that contain more than one nodes.

2. Find the Markov tree that is representative of the hypergraph by clustering variables, as outlined in Algorithm 4.3.2. First the maximal intersection graph (MIG) of the reduced hypergraph elements is constructed. This results in getting all possible combinations between the reduced-hypergraph elements that are adjacent to each other, containing common nodes from the original Bayesian network. This MIG guarantees considering all possible combinations of the intermediate Bayesian network nodes that can influence each other. The MIG is then combined with the subset of hyperedges removed during the reduction of the original hypergraph, and the result is simplified to remove duplicates and redundancies (i.e. node sets that are strict subsets to other sets). Finally, the singular node sets are also added to complete the Markov tree.

This completes the first stage of the construction of the join tree for the subsequent reasoning. The reasoning starts by determining the nodes for which evidence is provided. Using the principles in the description of the message passing phase above, the evidence is combined with each join tree valuation moving toward the root and leaf nodes of the tree. Once evidence is propagated the collection of the effect that the evidence had on the valuations of individual nodes in the original Bayesian network, i.e. the posterior selection probabilities for the individual model fragments, is performed by marginalising the updated probability distribution for the entire network for individual variables represented by the Bayesian network nodes. This provides the final probabilities of every model fragment being selected or rejected.

ALGORITHM 4.3.2: GENERATE_MARKOV_TREE(*relations*)

```
$r \leftarrow$ REDUCE_HYPERGRAPH($relations$)
if $r \equiv \emptyset$ then
    SIMPLIFY_TREE()
    ADD_NODES_TO_MARKOV(ALL_VERTICES($relations$))
else
    comment: The graph could not be reduced all the way
    comment: Construct maximal intersection graph (MIG) with what is left
    $g \leftarrow$ CONSTRUCT_MIG($r$)
    comment: Produce a Markov tree embedding from the MIG
    MAKE_MARKOV_TREE($g$)
    SIMPLIFY_TREE()
    ADD_NODES_TO_MARKOV(ALL_VERTICES($relations$))
```

4.4 Model Formulation Output

The result of the model formulation process is a set of probabilities for the selection or rejection of each model fragment for all components in the domain system. The fragment that will represent a component in the final model is the one with the highest selection probability among all fragments in the corresponding component assumption class.

The result set of model fragments that are selected for representing the corresponding components in the formulated model, is guaranteed to be complete, in terms of containing a fragment for every component in the domain system examined. Furthermore, the information needs of the user with regard to those components that user queries target are met either exactly, if the user specifically requested a resolution level, or approximately based solely on resolution preference from previous interactions. The former case is guaranteed based on the design of the Approximate User Monitor (AUM), which determines the resolutions of the components that are the targets of user queries to be exactly what the user requested. The latter case, when there is no such specific request, though it does not provide similar guarantees that it will result in AUM recommending an optimal model fragment resolution, it guarantees that the resolution will be based on the aggregation of indications from previous interactions of what the user information needs are, and the AUM's rules that are critical in this decision making can be calibrated experimentally to reach optimal recommendations for a variety of users.

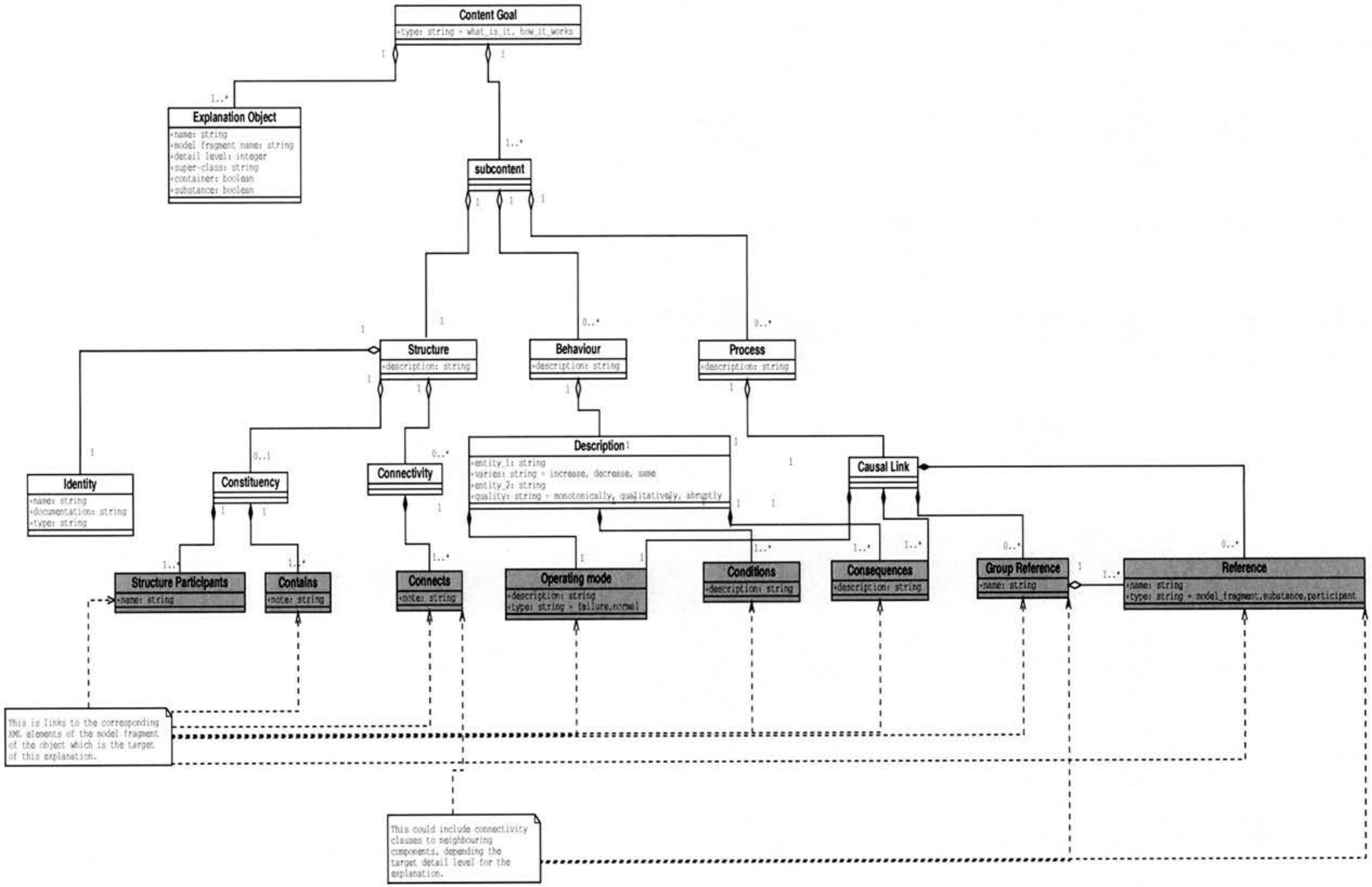
For the remaining components, which have no previous indication from either the user or the AUM on what their resolution level should be, the following guarantee can also be made. The use of Bayesian networks can guarantee that evidence coming from the AUM will be combined with previous indications on the possible selection influences between fragments of alternative components, to result in recommendations that are compliant to both of these sources of information (evidence and prior probabilities). However, in no way there is a guarantee that the final model fragments will constitute a correct mathematically model. The framework does not attempt to answer this specific question, only targets the provision of a methodology for handling user information requirements. In fact the assumption in this thesis is that the model fragments themselves are previously verified via other processes not described herein with regard to their individual correctness.

A model is finally constructed by combining the selected model fragments through their terminal connections, according to the initial description of the domain system structure in the given scenario. The final model constituents, in actual implementation, are the XML representations of the model fragments that were selected, and XML descriptions of the connectivity between individual fragments, which is an XML description of the scenario connectivity statements. After the combination of model fragments, those variables that receive no causal influence from other model variables are defined as model "exogenous", i.e. they are determined by factors outside the model's boundary, thus defining the scope of the model (Leitch et al., 1999). The resulting model can be subsequently used, after an analysis of its contents, to extract possible information that can be communicated to the user.

The implementation of this analysis consists of mainly parsing the XML description of the formulated model, to create a content description of the explanation being prepared with all the information that is required to cover the detail requested from the user. The parser takes first into account the range of valid XML elements that can be contained in a valid XML explanation-content description file. The validity of these kind of XML files is described inside the content **document type definition** (or DTD) file, an illustration of which is shown in Figure 4.6. In particular, for each type of explanation goal, as formed by the initial user question, a content goal is constructed bearing the type definition for the target explanation goal. Given the model representation from the Model Formulation module, the parser then extracts information from the model fragment or fragments involved in the user query:

"What is it" type of questions The content for this type of questions includes the description of the identity of the component that is explained, its constituency in terms of the list of participants appearing in its corresponding model fragment as well as its connectivity both internally between the different participants listed and externally to its closest neighbour components. All this information is part of the model fragment representations in the formulated model. The amount of detail that can be extracted is determined by the resolution level of the model fragment selected for the examined component, as the parser also considers the resolution level as an indication of what needs to be extracted. For novices the parser seeks to extract basic information regarding the consistency and basic connectivity of the model fragment in question. For intermediate users, on the other hand, the parser attempts also to determine connections between the fragment for the component in question and fragments for the nearest neighbouring components. Finally, expert users also get an indication of which participants are related to the different operating modes of the component, for normal and failure modes.

"How does it work" type of questions This type of query requires that the parser extracts information regarding the way the internals of the examined component



This is links to the corresponding XML elements of the model fragment of the object which is the target of this explanation.

This could include connectivity clauses to neighbouring components, depending the target detail level for the explanation.

Figure 4.6: XML DTD associations used by the Content Manager.

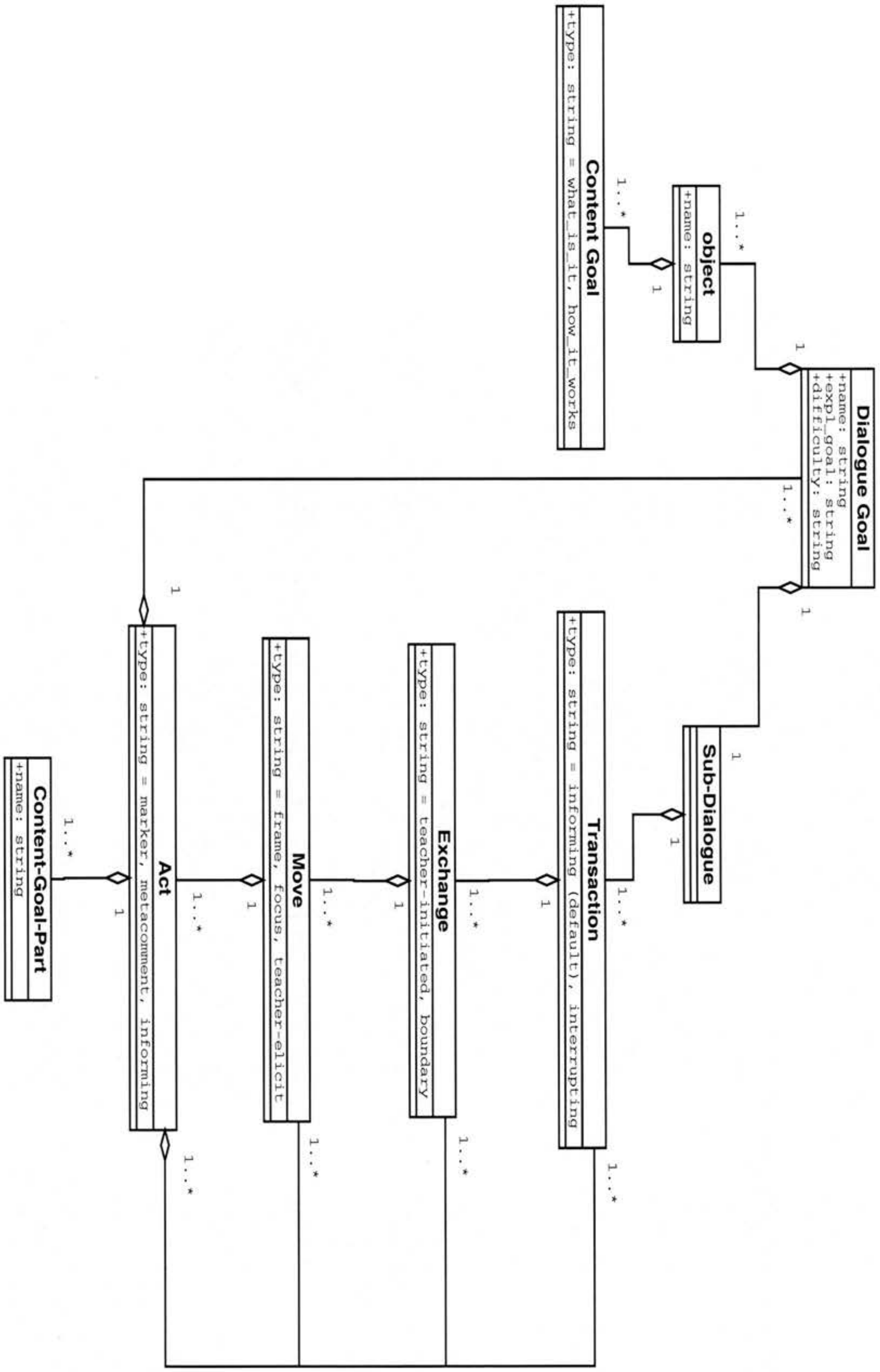

Content Goal
+type: string = what_is_it, how_it_works
object
+name: string
Dialogue Goal
+name: string
+expl_goal: string
+difficulty: string
Sub-Dialogue
Transaction
+type: string = informing (default), interrupting
Exchange
+type: string = teacher-initiated, boundary
Move
+type: string = frame, focus, teacher-elicit
Act
+type: string = marker, metacomment, informing
Content-Goal-Part
+name: string
1..*
1
1..*
1
1..*
1
1
1
1..*
1
1..*
1
1..*
1
1..*
1
1..*
1..*
1..*
1..*

Figure 4.7: XML DTD associations used by the Dialogue Manager.

work for novice users, its behaviour with respect to neighbouring components (for its normal mode operation) for intermediate user types, and the different ways in which the component can fail for the experts. To reiterate, all this is information that is contained in the model fragments. The internal workings are explained based on the connectivity of the participants of the fragment that represents the component of interest. The descriptions of the fragment's equations in its consequences it can be used to express the ways the internal participants behave.

The question types handled in this work were selected based on intuition, whilst the methodology of tackling each question type, as outlined in the previous list was partly motivated from available research regarding user expectations in generated explanation content — most notably the work in Cawsey (1991). Thus, the rationale behind the choice of question types and the selected manner for handling these types should be regarded as an intuitive and plausible, yet untested, treatment.

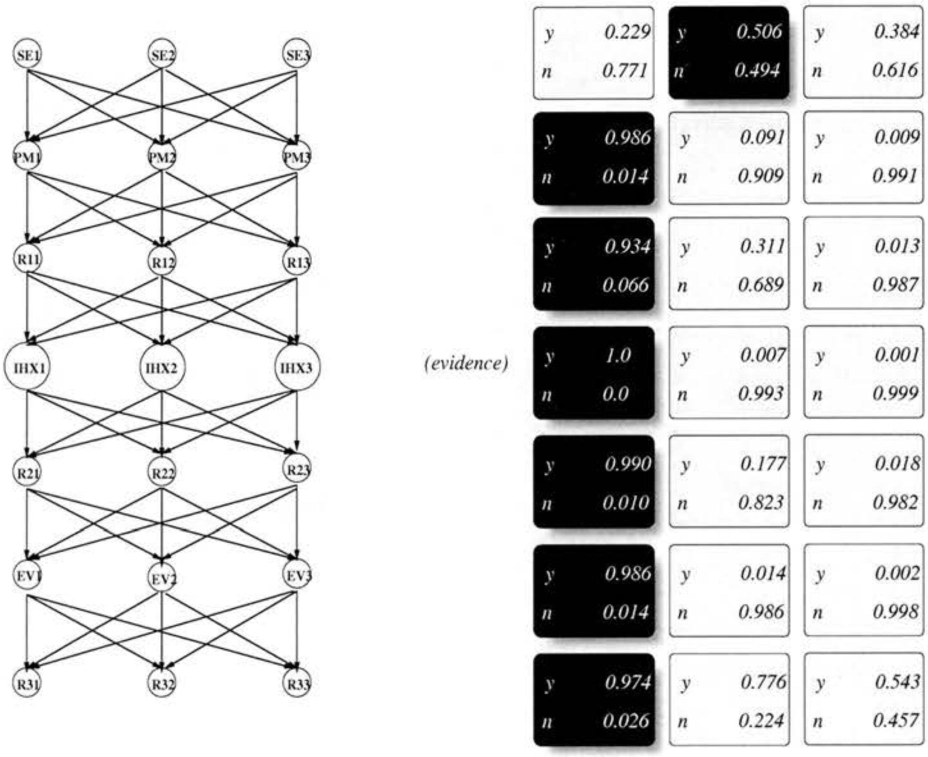
When the explanation content is prepared the Dialogue Manager can identify the way that this content can be conveyed to the user. The general guidelines on how this can be done are listed draw from the work described in (Cawsey, 1991). This thesis does not focus into contributing towards a theory of conveying explanation content to users. Nevertheless, the Dialogue Manager builds the information regarding this communication, using XML to identify relationships between concepts that are important for this communication. These concepts include whether the explanation is a teaching transaction, i.e. a transaction as defined in 2.2.3 and (Cawsey, 1991) that is initiated by the system, or whether it comes as a response to a direct question, what type of utterances to use around the explanation content introduction etc. The XML used for this purpose is built based on the Dialogue Manager DTD which is depicted in Figure 4.7. Prior to generating the actual explanation text that is conveyed to the user, the representation of an explanation by the Dialogue Manager consists of a set of 'Dialogue Goal' XML structures, each of which, in turn, consists of a set of XML 'content goal' structures, which are used to determine the content of the explanation, a number of 'sub-dialogue' XML structures that determine how to utter this content. Similarly to the Content Manager, the Dialogue Manager uses the type of question asked as a way to determine which content XML constructs to use. As for the manner to convey the content, the Dialogue Manager decides this based on the previous discourse record with the user, once again based on guidelines from (Cawsey, 1991). The result of this process is a set of dialogue goal XML structures that aim to present the explanation content to the user.

The following sections provide examples for the model formulation process that has been outlined thus far, providing additional information about the usage of the results of the model formulation process by the explanation generation subprocess.

4.4.1 Scenario 1: Explaining a Single Component

Consider a user that has just started interacting with the explanatory system, asking "what-is-it" for the IHX component of Figure 3.4. Since there has been no previous entry in the history of discourse which is related to this query, a new explanation goal is formed to be what_is_it$\{IHX\}$. The profiles currently stored in the User Monitor for this user are empty and thus the Approximate Reasoner determines the user to be "novice" and the desired detail for the model fragment that represents the component of interest to be "low". The resulting explanation goal, the objects of interest and the (crisp, not fuzzy) value of the resolution for the IHX component are added to the User Monitor (for future use). The decision about the resolution of the model fragment for IHX, which is available in the fragment library, is provided as evidence to the Bayesian network of the Model Formulator (should there be no fragment conforming to the resolution level requirement the fragment with the next higher resolution level would be picked instead, whilst the user would be notified that the model fragments library is incomplete).

The Bayesian network is subsequently used with the given evidence, to determine the best choice of model fragments for the remaining components. The selection of fragments is illustrated in Figure 4.8, with the shaded tables indicating the selected fragments for the corresponding components. Each of these selected fragments has the highest probability to be chosen among all fragments organised under the



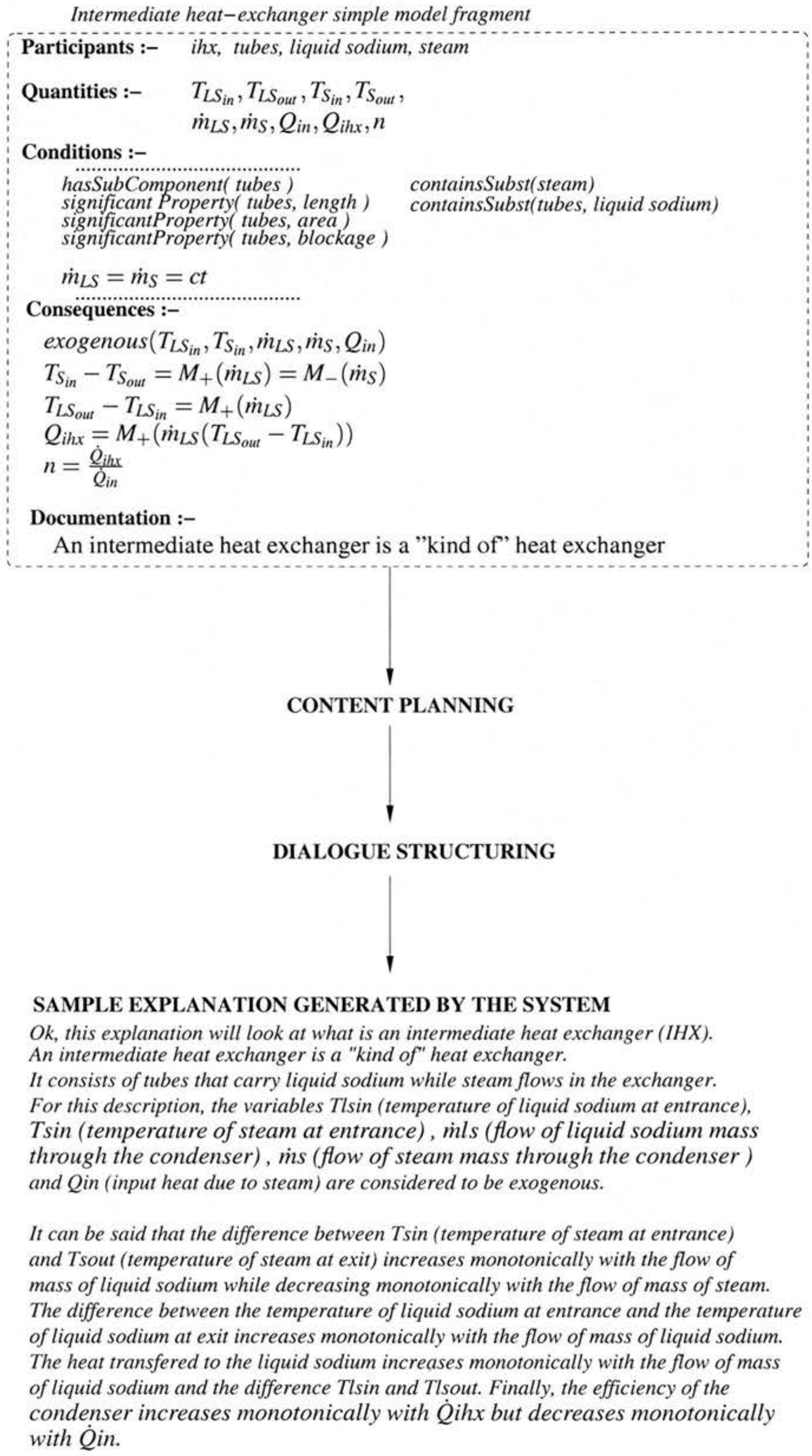
	y 0.229 n 0.771	y 0.506 n 0.494	y 0.384 n 0.616
	y 0.986 n 0.014	y 0.091 n 0.909	y 0.009 n 0.991
	y 0.934 n 0.066	y 0.311 n 0.689	y 0.013 n 0.987
(evidence)	y 1.0 n 0.0	y 0.007 n 0.993	y 0.001 n 0.999
	y 0.990 n 0.010	y 0.177 n 0.823	y 0.018 n 0.982
	y 0.986 n 0.014	y 0.014 n 0.986	y 0.002 n 0.998
	y 0.974 n 0.026	y 0.776 n 0.224	y 0.543 n 0.457

Figure 4.8: Bayesian model fragment selection for scenario 1.

same assumption class of the corresponding component (a more interesting sample for Bayesian network aided updates of fragment selection will be given in the next sub-section). The final model is composed by "connecting" the fragments via their terminal variables.

The resultant model is used by the Content Manager module to extract information to be communicated to the user. This is done by parsing the fragments participating in the model, extracting information regarding the IHX's structure and (for the current, non-expert user) general issues about IHX's function. The result of this parsing is a piece of XML code with the textual extracts from the mode fragments encapsulated within XML tags, such as the XML excerpt shown in Figure 4.9. For presentational simplicity, the XML excerpts are shown only for the case of explaining the simple fragment of IHX. It consists mainly of tags for the structural details of the IHX component plus descriptions of the qualitative equations contained in the model fragment selected for IHX.

The extracted information bears no embedded indications about how it should be conveyed to the user. This is a task for the Dialogue Manager module, which receives the file containing the XML extracts from the Content Manager and uses it to generate another XML file, a plan for communicating the explanation content to the user. These two XML files are subsequently parsed, resulting in an HTML page showing the explanation regarding the question asked by the user, i.e. "what is an IHX". The part of the explanation generated to satisfy the user's query about IHX is illustrated in Figure 4.9 (explanation content, generated for a novice, to appear on screen). Samples of the XML generated during the content planning and the dialogue structuring stages are demonstrated in Figures 4.10 and 4.11. When the explanation is finalised and displayed, the discourse history is updated with the Dialogue Manager's explanation plan.



Intermediate heat–exchanger simple model fragment

Participants :– *ihx, tubes, liquid sodium, steam*

Quantities :– $T_{LS_{in}}, T_{LS_{out}}, T_{S_{in}}, T_{S_{out}},$
$\dot{m}_{LS}, \dot{m}_S, Q_{in}, Q_{ihx}, n$

Conditions :–

hasSubComponent(tubes)
significant Property(tubes, length)
significantProperty(tubes, area)
significantProperty(tubes, blockage)

containsSubst(steam)
containsSubst(tubes, liquid sodium)

$\dot{m}_{LS} = \dot{m}_S = ct$

Consequences :–

$exogenous(T_{LS_{in}}, T_{S_{in}}, \dot{m}_{LS}, \dot{m}_S, Q_{in})$
$T_{S_{in}} - T_{S_{out}} = M_+(\dot{m}_{LS}) = M_-(\dot{m}_S)$
$T_{LS_{out}} - T_{LS_{in}} = M_+(\dot{m}_{LS})$
$Q_{ihx} = M_+(\dot{m}_{LS}(T_{LS_{out}} - T_{LS_{in}}))$
$n = \frac{\dot{Q}_{ihx}}{\dot{Q}_{in}}$

Documentation :–
An intermediate heat exchanger is a "kind of" heat exchanger

CONTENT PLANNING

DIALOGUE STRUCTURING

SAMPLE EXPLANATION GENERATED BY THE SYSTEM

Ok, this explanation will look at what is an intermediate heat exchanger (IHX). An intermediate heat exchanger is a "kind of" heat exchanger. It consists of tubes that carry liquid sodium while steam flows in the exchanger. For this description, the variables Tlsin (temperature of liquid sodium at entrance), Tsin (temperature of steam at entrance) , ṁls (flow of liquid sodium mass through the condenser) , ṁs (flow of steam mass through the condenser) and Qin (input heat due to steam) are considered to be exogenous.

It can be said that the difference between Tsin (temperature of steam at entrance) and Tsout (temperature of steam at exit) increases monotonically with the flow of mass of liquid sodium while decreasing monotonically with the flow of mass of steam. The difference between the temperature of liquid sodium at entrance and the temperature of liquid sodium at exit increases monotonically with the flow of mass of liquid sodium. The heat transfered to the liquid sodium increases monotonically with the flow of mass of liquid sodium and the difference Tlsin and Tlsout. Finally, the efficiency of the condenser increases monotonically with Q̇ihx but decreases monotonically with Q̇in.

Figure 4.9: Extracting information from model fragments via content and dialogue planning.

CONTENT PLANNING

```
<!ELEMENT c-goal (object)+ subcontent>     <!ATTLIST c-goal name CDATA #REQUIRED
                                                     expl_goal CDATA #REQUIRED
                                                     difficulty  CDATA #REQUIRED>

<!ELEMENT object>        <!ATTLIST object name CDATA #REQUIRED
                                          sym_name CDATA #REQUIRED
                                          detail_level  CDATA #REQUIRED
                                          super_class  CDATA #REQUIRED
                                          container (yes | no ) "no"
                                          substance (yes | no ) "no">


<!ELEMENT subcontent structure (behaviour | process)*>
<!ELEMENT structure identity (constituency)? (connectivity)*>
<!ELEMENT behaviour (description)+>
<!ELEMENT process (causal_sequence)+>

 ..........
```

CONTENT DTD

```
<c-goal name = "what is a IHX" expl_goal="what_is_it(IHX)" difficulty="simple">
 <object name = "intermediate heat exchanger" sym_name="IHX" detail_level="1"
          super_class="heat exchanger" container="yes" substance="no">
    <object name="pipes" .......>
      <object name="Cooling water" ........ >
      </object>
    </object>
    <object name = "steam" .... ></object>
 </object>
  ..........
 <subcontent>
   <structure>
     <identity object_name="Intermediate Heat Exchanger"  type="kindof" object_name="heat exch."/>
     <behaviour object_name="IHX"mode_of_operation="normal">
       <description entity_1="difference between Tsin and Tsout"
                    varies="increase" quality="monotonically"
                    entity_2="flow of CW mass"/>
       <description entity_1="difference between Tcwout and Tcwin"
                    varies="increase" quality="monotonically"
                    entity_2="flow of CW mass"/>
      <description entity_1="Qtr"
                   varies="increase" quality="monotonically"
   entity_2="product between flow of CW mass and difference between Tcwout and Tcwin/>
   </behaviour>
   </subcontent>
   </c-goal>
```

CONTENT XML

Figure 4.10: Extracting information from model fragments: XML excerpts for content planning.

DIALOGUE STRUCTURING

```
<!ELEMENT d-goal (object)+ subdialogue>          <!ATTLIST d-goal name CDATA #REQUIRED
                                                              expl_goal CDATA #REQUIRED
                                                              difficulty  CDATA #REQUIRED>
<!ELEMENT object c-goal_name+> <!ATTLIST object name CDATA #REQUIRED>
<!ELEMENT subdialogue transaction+>
<!ELEMENT transaction exchange+>  <!ATTLIST transaction type (informing | interrupting) informing>
<!ELEMENT exchange move+>  <!ATTLIST exchange type (teacher-initiated | boundary) >
<!ELEMENT move act+>  <!ATTLIST move type (frame | focus | teacher-elicit)>
<!ELEMENT act (c-goal-part | d-goal | transaction | exchange | move)+>
     <!ATTLIST act type (marker | metacomment | informing) >
<!ELEMENT c-goal-part (EMPTY)><!ATTLIST name-of-part CDATA #REQUIRED>


 ..........
```

DIALOGUE DTD

```
<d-goal name="d-goal for IHX" expl_goal="what_is_it(IHX)" difficulty="simple">
 <c-goal name="what is a IHX"/>
   <transaction type="informing">
      <exchange type="boundary">
         <move type="framing">
            <act type="marker"></act>
         </move>
        <move type="focusing">
           <act type="metacomment"></act>
        </move>
      </exchange>
     <exchange type="teacher-initiated">
         <move type="teacher-elicit">
           <act type="informing"><c-goal-part name-of-part="structure"></act>
           <act type="informing"><c-goal-part name-of-part="behaviour"></act>
         </move>
     </exchange>
     <exchange type="boundary">
        <move type="closing">
          <act type="closing">
          </act>
        </move>
     </exchange>
  </transaction>
 </subdialogue>
</d-goal>
```

DIALOGUE PLAN XML

Figure 4.11: Extracting information from model fragments: XML excerpts for dialogue planning.

4.4.2 Scenario 2: Bayesian Network Fragment Selection for Multiple Components

In this scenario, assumptions are made about the initial setting of the user's interaction with the system, none of which incurs any cost to the generality of this example. First, it is assumed that the user has already had some interaction with the system, resulting in explanations generated for the components: P/M (pump/motor component) and EV (evaporator). The user's expertise as modelled by the Approximate User Monitor has also been revised for those components, so that the user has reached the levels of "intermediate" and "expert" for P/M and EV respectively. Furthermore, it is assumed that there have been no previous references to the IHX component (intermediate heat exchanger). The final assumption places the user at the point when he/she is about to ask "what is an intermediate heat exchanger".

After formulating the explanation goal for this user request, `what_is_it`$\{IHX\}$, the Approximate User Monitor (based on the assumptions made above) determines that the detail level for the representation of the IHX component should be "low". The detail levels for P/M and EV remain unchanged, that is, "medium" and "high" respectively, as these objects do not participate in the explanation goal. Therefore, the evidence passed to the Model Formulator is, "medium" detail for P/M, "high" detail for EV and "low" detail for IHX. The result of the Bayesian network reasoning is shown in case *A* of Figure 4.12. The components for which evidence is provided by the Approximate User Monitor are marked in the figure, and the selected fragments are again denoted by the dark shaded arrays.

Assuming that the user continues asking information on how the *IHX* component works, the Approximate User Monitor indicates that an intermediate detail fragment should be used, while the detail levels for P/M and EV remain unchanged. With this evidence, the Bayesian network performs the fragment selection as indicated in case *B* of Figure 4.12. If, after updating the profiles and expertise level of the user, further information is required on how *IHX* works, the Approximate Reasoner indicates that a most detailed representation of the particular component should be used. The selection pattern of fragments as completed by the Bayesian Network is then as illustrated

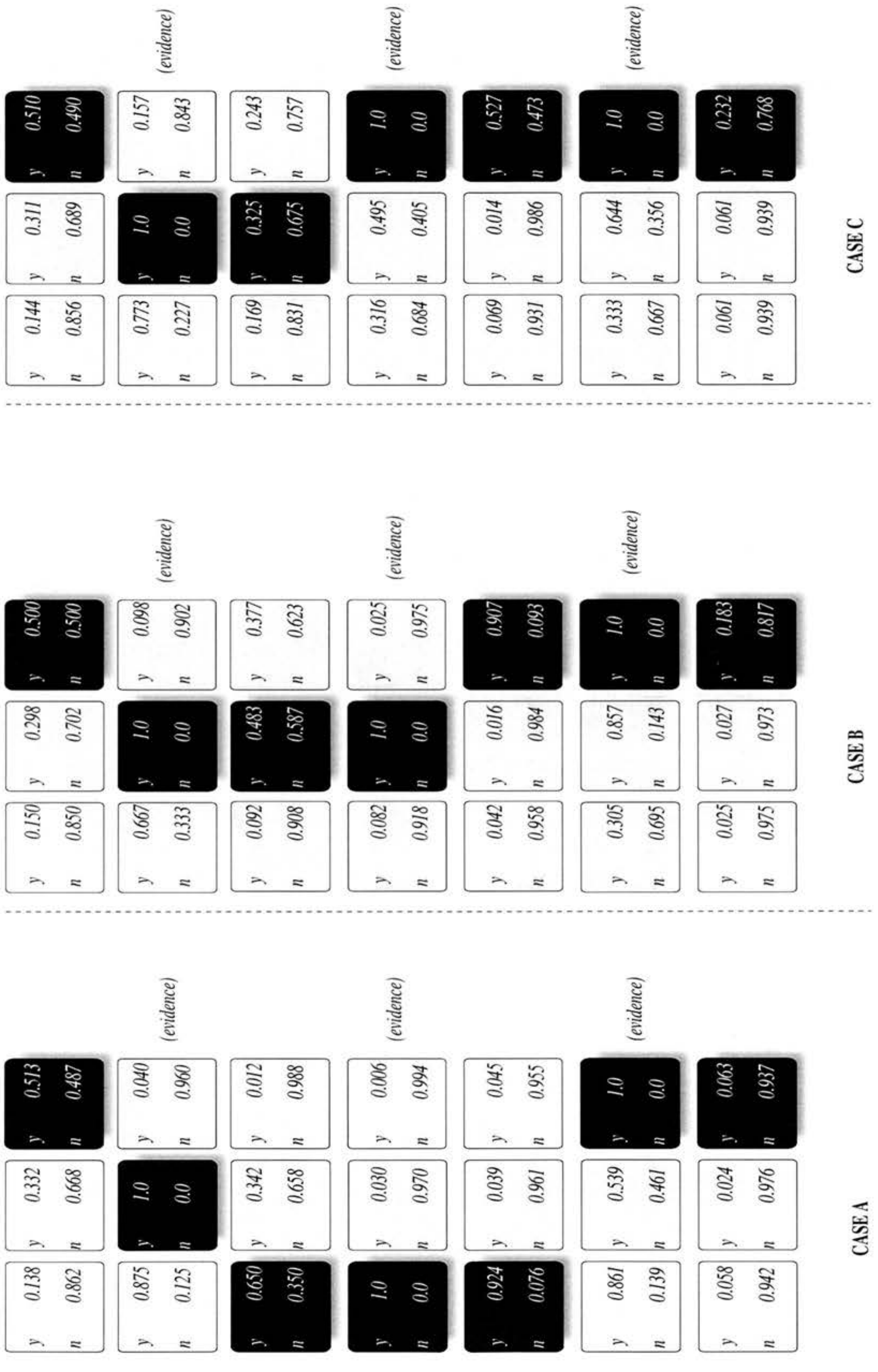


Figure 4.12: Model fragment selection: gradual increase of detail requirements.

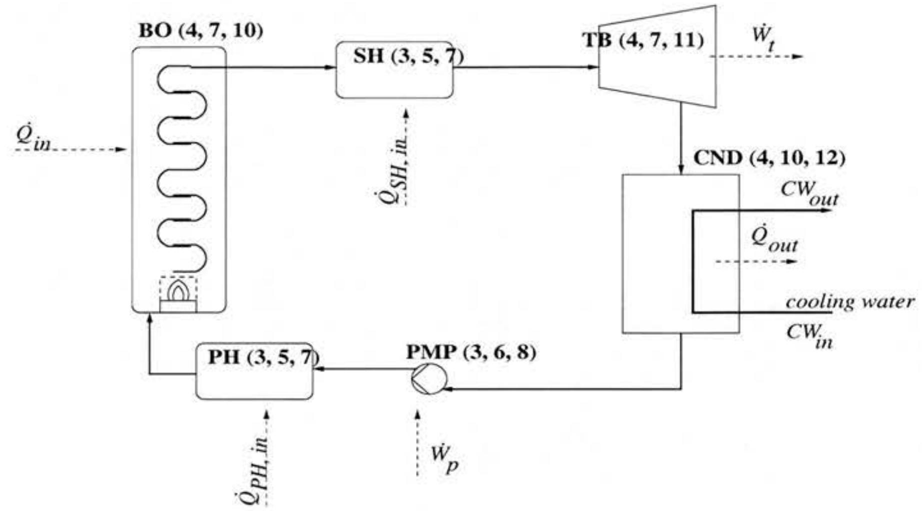
in case *C* of Figure 4.12. By comparing cases *B* and *C* to *A*, it can be seen that the Bayesian network amends its proposal for selecting suitable fragments for the remaining components according to the incoming evidence. In all cases, the detail level of the fragments for those components that are not supported by evidence is regulated by the detail levels decided for their adjacent components. That is, the Bayesian network attempts to map as closely as possible the detail levels of the fragments selected for neighbouring components onto the required levels of undecided component fragments, in addition to trying to conform to the provided evidence. This is quite intuitive and the results obtained seem to match well with what is required to generate adequate explanations.

4.5 Experiments with the Rankine Cycle

The present section outlines experiments carried out with the model fragment selection algorithm, in a different domain system than the one used in the examples listed in Sections 4.4.1 and 4.4.2. In the following paragraphs the domain is described, as well as the Bayesian network used for the selection process and examples of the resulting explanations for user interaction scenarios, similar to those described in Sections 4.4.1 and 4.4.2

The domain system used for these experiments is the set of thermodynamic phenomena in a simple Rankine cycle process of steam production to generate energy for driving a turbine, followed by steam cooling, and collection of the water condensate produced during the cooling process for recycling through the steam production stage (Moran and Shapiro, 1992). This is a rather well studied and understood industrial process in engineering thermodynamics providing a good background to facilitate the elicitation of model fragment representations for the purpose of explaining the components involved.

A simple illustration of the components involved in the Rankine cycle is given in Figure 4.13.



Notation guide: e.g. BO (4, 7, 10): BO denotes the boiler component name, while the triplet (4, 7, 10) denotes the resolutions of the boiler fragments

Figure 4.13: The Rankine thermodynamic cycle: components connections and model fragment resolutions.

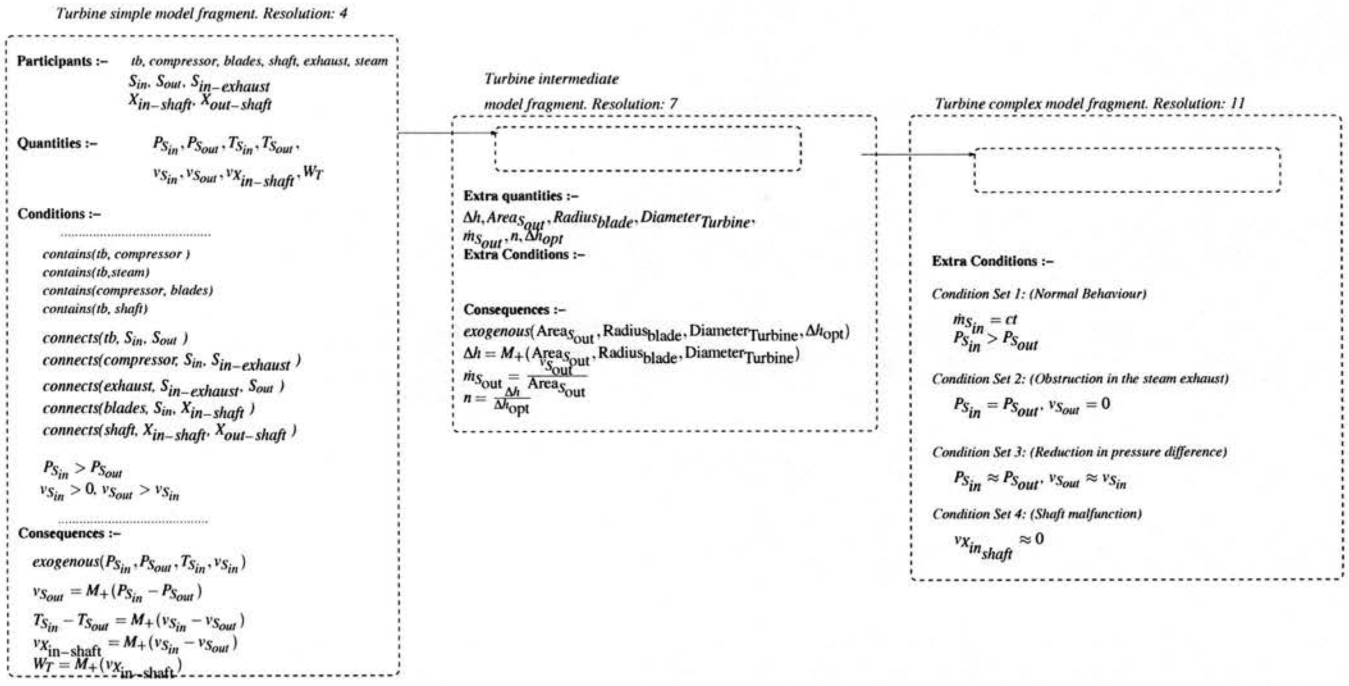
As described in (Moran and Shapiro, 1992), the Rankine cycle is one of the simplest thermodynamic cycles used by power-generation plants. A power plant representation consists of a boiler (BO), turbine (TB), condenser (CND), a pump (PMP), pre-heater (PH) and a super-heater (SH), as indicated in Figure 4.13. Fuel, burned in the boiler, heats the water which has been already preheated in the pre-heater under the pressure environment conditions created by the pump, generating steam. The steam is further heated in an isobaric process in the super-heater, thus generating superheated saturated steam. This saturated steam is used to rotate the turbine which powers a generator. Flowing through the turbine the steam goes through an iso-entropic expansion process, i.e. its temperature decreases while its volume increases under the assumption that there is no heat exchange with the outer ambient. Electrical energy is generated when the generator windings, connected to the turbine via a shaft, rotate in a strong magnetic field. After the steam leaves the turbine, it is cooled to its liquid state in the condenser, in an isobaric process keeping the same pressure as in the turbine. The liquid is pressurised by the pump, prior to going back through the heating stages. The Rankine cycle describes the physical processes that take place as steam circulates in the various components of the plant, and provides the mathematical principles to calculate the net energy gain in terms of energy generated by the turbine. It could be combined with the system used so far, the liquid sodium cooling loop as illustrated in Figure 3.4, which could become part of the steam generation sub-process of the Rankine cycle serving as the pre-heater component of the cycle shown in Figure 4.13.

For simplicity in the scenario description for this domain system only the components appearing in the Figure 4.13 are considered without accounting for any connection 'resistance' between the components involved, as was done for the example liquid-sodium cooling system used throughout this chapter. Though simplifying, this assumption is not in anyway demeaning the resulting observations for the model fragment selection or the type of explanation provided — there simply are some components that do not appear in the explanations. Additionally, if the model fragments developer so wishes, it would be possible to embed connection resistances in the descriptions of the consequences included each of the model fragments used for the corresponding components.

Once again, 3 model fragments are considered for each of the components in this system, following the general guidelines provided earlier in this chapter (Section 4.1.1). The model fragments for the component TB of the Rankine cycle can be seen in Figure 4.14. The scenario and corresponding Bayesian network, as well as the tables of prior probabilities for the network nodes related to the turbine (TB) component are illustrated in Figure 4.15.

Given the scenario, the Bayesian network, and the prior probabilities for the network nodes it is now possible to run some experiments and examine the selected model fragments and the expected explanations. A sample of the experimental runs of the program will be presented below, each accompanied with an illustration of the final Bayesian network model fragment selection and the explanation.

Three instances of the user interaction with the program are considered for requests on "how a turbine works", covering the cases where the user is classified as a "novice", "novice" or "expert". The first instance, the end result of which is shown in Figure 4.16, occurs directly at the start of the user's interaction, after requesting information on the condenser component using an intermediate level of detail for the condenser.



Turbine simple model fragment. Resolution: 4

Participants :– tb, *compressor, blades, shaft, exhaust, steam*
$S_{in}, S_{out}, S_{in-exhaust}$
$X_{in-shaft}, X_{out-shaft}$

Quantities :– $P_{S_{in}}, P_{S_{out}}, T_{S_{in}}, T_{S_{out}},$
$v_{S_{in}}, v_{S_{out}}, v_{X_{in-shaft}}, W_T$

Conditions :–

contains(tb, compressor)
contains(tb,steam)
contains(compressor, blades)
contains(tb, shaft)

connects(tb, S_{in}, S_{out})
connects$(compressor, S_{in}, S_{in-exhaust})$
connects$(exhaust, S_{in-exhaust}, S_{out})$
connects$(blades, S_{in}, X_{in-shaft})$
connects$(shaft, X_{in-shaft}, X_{out-shaft})$

$P_{S_{in}} > P_{S_{out}}$
$v_{S_{in}} > 0, v_{S_{out}} > v_{S_{in}}$

Consequences :–

$exogenous(P_{S_{in}}, P_{S_{out}}, T_{S_{in}}, v_{S_{in}})$
$v_{S_{out}} = M_+(P_{S_{in}} - P_{S_{out}})$
$T_{S_{in}} - T_{S_{out}} = M_+(v_{S_{in}} - v_{S_{out}})$
$v_{X_{\text{in-shaft}}} = M_+(v_{S_{in}} - v_{S_{out}})$
$W_T = M_+(v_{X_{\text{in-shaft}}})$

Turbine intermediate model fragment. Resolution: 7

Extra quantities :–
$\Delta h, Areas_{S_{out}}, Radius_{blade}, Diameter_{Turbine},$
$\dot{m}_{S_{out}}, n, \Delta h_{opt}$
Extra Conditions :–

Consequences :–
$exogenous(\text{Areas}_{S_{\text{out}}}, \text{Radius}_{\text{blade}}, \text{Diameter}_{\text{Turbine}}, \Delta h_{\text{opt}})$
$\Delta h = M_+(\text{Areas}_{S_{\text{out}}}, \text{Radius}_{\text{blade}}, \text{Diameter}_{\text{Turbine}})$
$\dot{m}_{S_{\text{out}}} = \frac{v_{S_{\text{out}}}}{\text{Areas}_{S_{\text{out}}}}$
$n = \frac{\Delta h}{\Delta h_{\text{opt}}}$

Turbine complex model fragment. Resolution: 11

Extra Conditions :–

Condition Set 1: (Normal Behaviour)
$\dot{m}_{S_{in}} = ct$
$P_{S_{in}} > P_{S_{out}}$

Condition Set 2: (Obstruction in the steam exhaust)
$P_{S_{in}} = P_{S_{out}}, v_{S_{out}} = 0$

Condition Set 3: (Reduction in pressure difference)
$P_{S_{in}} \approx P_{S_{out}}, v_{S_{out}} \approx v_{S_{in}}$

Condition Set 4: (Shaft malfunction)
$v_{X_{in_{shaft}}} \approx 0$

Legend:

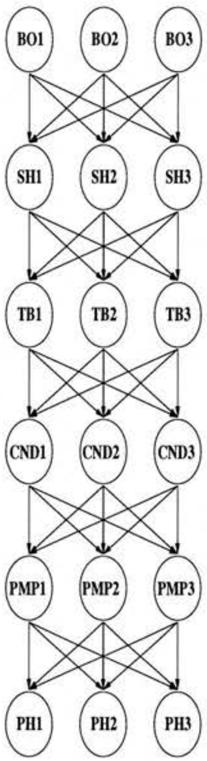
S_{in}, S_{out} : Terminals for heated steam circulation in the TB,
$S_{\text{in-exhaust}}$: Terminal for steam input to exhaust of TB (past blades)
$X_{\text{in-shaft}}, X_{\text{out-shaft}}$: Terminals for connection of the TB shaft
$P_{S_{in}}, P_{S_{out}}$: Pressure of steam at inlet/outlet terminals
$T_{S_{in}}, T_{S_{out}}$: Temperature of steam at inlet/out;et terminals
$v_{S_{in}}, v_{S_{out}}$: Steam flow speed at inlet and outlet terminals
$v_{x_{\text{in-shaft}}}$: speed of steam at the shaft connection terminal
W_T : Total work produced by the TB

Δh : Enthalpy change between inlet and outlet terminals
$\text{Area}_{S_{out}}$: Cross-section of the steam outlet terminal
$\text{Radius}_{\text{blade}}$: Turbine blade radius
$\text{Diameter}_{\text{Turbine}}$: Total turbine diameter
$\dot{m}_{S_{out}}$: Flow of steam at steam outlet terminal
n : Turbine efficiency
Δh_{opt} : Optimal enthalpy change

Figure 4.14: Model fragments excerpt for the component Turbine (TB).

Scenario for Rankine cycle:

boiler(BO)
hydraulic_connection(BO, X6, X1)
superheater(SH)
hydraulic_connection(SH, X1, X2)
turbine(TB)
hydraulic_connection(TB, X2, X3)
mechanical_connection(TB, X2, X3)
condenser(CND)
hydraulic_connection(CND, X3, X4)
pump(PMP)
hydraulic_connection(PMP, X4, X5)
preheater(PH)
hydraulic_connection(PH, X5, X6)



	T00	
BO1	y	0.33
	n	0.67

	T01	
BO2	y	0.33
	n	0.67

	T02	
BO3	y	0.33
	n	0.67

Prior probabilities of selecting a model fragment for component "BO"

SH1 & SH2 & SH3

	T11	yyy	yyn	yny	ynn	nyy	nyn	nny	nnn
TB1	y	0.0010	0.0098	0.0019	0.0244	0.0017	0.0219	0.0031	0.1040
	n	0.1040	0.1040	0.1040	0.1040	0.1040	0.1040	0.1040	0.1040

SH1 & SH2 & SH3

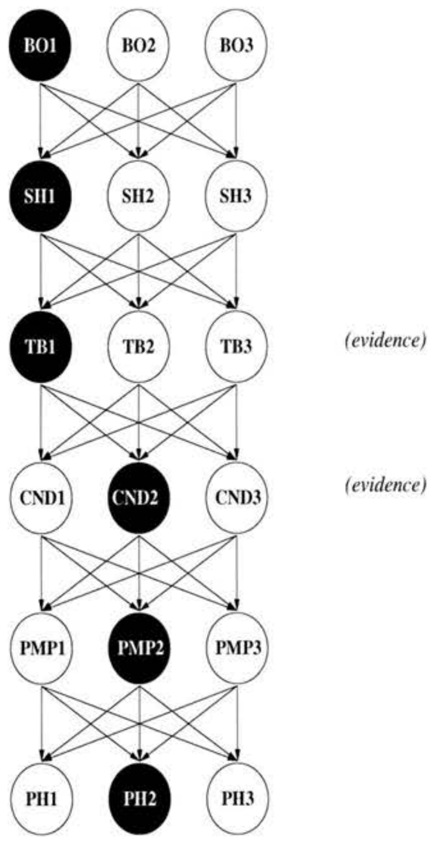
	T12	yyy	yyn	yny	ynn	nyy	nyn	nny	nnn
TB2	y	0.0007	0.0015	0.0001	0.0013	0.0041	0.0097	0.0345	0.1050
	n	0.1050	0.1050	0.1050	0.1050	0.1050	0.1050	0.1050	0.1050

SH1 & SH2 & SH3

	T13	yyy	yyn	yny	ynn	nyy	nyn	nny	nnn
TB3	y	0.0000	0.0000	0.0000	0.0002	0.0001	0.0003	0.0004	0.1110
	n	0.1110	0.1110	0.1110	0.1110	0.1110	0.1110	0.1110	0.1110

Unnormalised conditional probabilities for the model fragments "TB1" "TB2" "TB3" with respect to the selection or rejection of the combination of model fragments "SH1" "SH2" SH3"

Figure 4.15: Scenario, Bayesian network and some priors for the Rankine thermodynamic cycle.



SAMPLE EXPLANATION

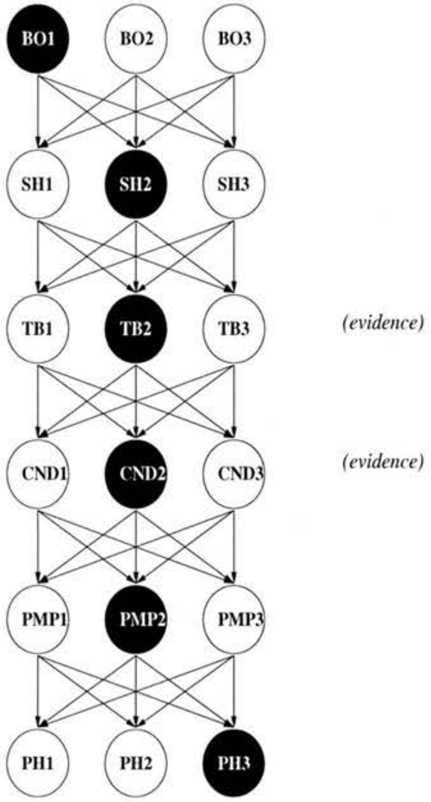
Ok, this explanation will look at how a turbine works (TB).

For this description, the variables PSin (pressure of steam at the turbine entrance), PSout (pressure of steam at the turbine exhaust) , TSin (steam temperature at the turbine entrance) and vsin (steam speed at the turbine entrance) are considered to be exogenous. It is also assumed that the steam pressure in the turbine exhaust is smaller than the pressure at the turbine entrance. It is also assumed that the steam velocity in the turbine entry is greater than zero, and the steam velocity at the turbine exhaust is greater than velocity at the turbine entry.

It can be said that the velocity of the steam at the turbine exit (vsout) increases monotonically as the difference in pressure between the turbine entry and exhaust increases. The steam output temperature also decreases monotonically as the steam output velocity increases. The speed of the shaft also increases as the output steam velocity increases. With the shaft velocity increasing, the produced work at the shaft also increases.

In the current system, a TB connects to PH (hyperlink) via CND (hyperlink) and to BO (hyperlink) via SH (hyperlink).

Figure 4.16: The Rankine thermodynamic cycle (from Figure 4.13): explaining how a turbine works to a novice.



SAMPLE EXPLANATION

OK, this explanation will look at how a turbine works.

For this description, the following quantities will be considered exogenous:
- *Steam pressure at the turbine entry (PSin)*
- *Steam pressure at the turbine exhaust (PSout)*
- *Steam temperature at the turbine entry (TSin)*
- *Steam velocity at the turbine entry (vSin)*
- *Exhaust outlet cross–section area.*
- *Turbine entry diameter*
- *Turbine blade radius*
- *Optimal enthalpy difference for the steam between the entry to the turbine the turbine exhaust outlet*

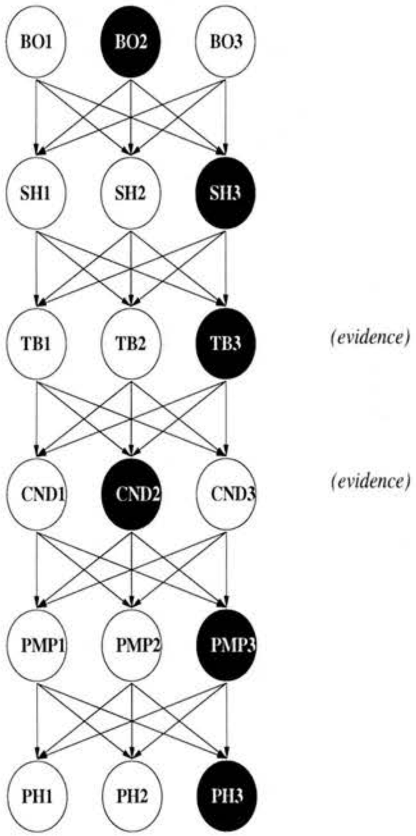
It is also assumed that PSin > PSout , vSin > 0 and vSout > vSin, where vSout. is the steam velocity at the turbine exhaust outlet.

It can be said that as the PSin – PSout difference increases vSout increases also (monotonically). As vSin – vSout increases, TSin – TSout decreases. Also, vXshaft (the rotational velocity of the shaft end) increases as vSout increases, and therefore the work produced at the end of the shaft (Wshaft) also increases.

Furthermore, the enthalpy difference in the steam passage through the turbine increases if the turbine diameter, the exhaust outlet cross–section area and the turbine blade radius increase. An increase of the enthalpy difference can cause increase to the turbine efficiency.

The turbine receives steam at temperature TSin and with velocity vSin from SH (a kind of heater), positioned after the boiler (BO). After the steam exists from the turbine, it passes to the condenser (CND) which is a kind of heat exchanger.

Figure 4.17: The Rankine thermodynamic cycle (from Figure 4.13): explaining how a turbine works with intermediate detail.



SAMPLE EXPLANATION

Ok, this explanation will look at how a turbine works (TB).

In particular, the various operating modes of the turbine (normal and failure) will be outlined.

For the turbine to operate normally, the condition is that there is a constant flow of steam through the turbine inlet, and that PSin > PSout, that is the steam pressure at the inlet of the turbine is greater than the steam pressure at the turbine exhaust.

If there is an obstruction in the steam exhaust, then PSin will become equal to PSout and vSout will be zero. This will cause the shaft velocity (vXshaft) to become also zero, and the work at the end of the shaft to become zero as well..

If there is a reduction in the pressure difference throughout the turbine (due to an obstruction in the super–heater SH) then PSin will be almost equal to PSout and vSout almost equal to vSin.

If there is a shaft malfunction then the velocity of the shaft can become almost zero, causing the decrease of the work produced Wshaft.

Figure 4.18: The Rankine thermodynamic cycle (from Figure 4.13): explaining how a turbine works to an expert.

The Approximate User Monitor (AUM) makes the suggestion, provided as evidence to the BN inference mechanism in the Model Formulator, to use *CND*2 and *TB*1 fragments. Since the user has just requested an explanation for *TB*, the AUM considers the user a "novice". The generated explanation is targeted for a novice, and thus the information regarding how the TB component works contains only a description of the equations described in the consequences of the fragment that was selected to represent TB.

In the second instance of this experimental run of the program, consideration is taken for the user's specific request for the TB fragment resolution to be intermediate, thus causing the AUM to suggest using the TB2 fragment. The resulting fragment selection by the BN and the explanation are illustrated in Figure 4.17. The AUM still considers the user to be a novice, since there have been too few queries targeted at this particular component. The resulting explanation is based on intermediate level of representation detail, but the content considers the user still to be a novice. The presence of sentences regarding what type of components are the *CND* and *SH*, the neighbouring components to *TB*, is an indication of the fact that the explanation has been generated for a novice. If the user was considered as "intermediate" in expertise by the AUM then the resulting explanation would not contain these extra descriptors.

Lastly, in the case of an expert user in turbines the AUM suggests the TB3 model fragment to be used, resulting in the network selection and explanation shown in Figure 4.18. Here the AUM already has determined that the user should be considered as "expert" in turbines, thus the resulting explanation is mainly an indication of the conditions involved in the different possible operating modes. The user is considered an "expert" based on previous interaction, which should have already covered the content in the explanation examples provided in the previous 2 instances of this experimental run. The content of these explanations is therefore not repeated. Only the possible operating modes are described via outlining the conditions that hold for the normal and failing operation of the turbine.

4.6 Comparison to Traditional CM Approaches

The general attributes of the CM approaches being compared can be set as in the attribute outline list below:

Causal Approximations (Nayak, 1994)

Task Explanation.

Primitives Model fragments organised in assumption classes. The fragments are ordered using simplicity and causal ordering criteria. Causal ordering determines how the fragment equations should be solved, thus altering the complexity of the equation satisfaction problem from NP to P.

Selection Process Models are composed in a "top-down" fashion: First the assumption classes that must be present are determined (by order of magnitude reasoning and the component interaction heuristic). Then the most complex fragment of each class is selected. Simplifications are applied to these fragments based on the concept of *causal approximations*.

Additional Refinement The simplification refinement process is an iterative process: simplifications are applied continuously on the selected fragments until no further useful explanation can be given based on the altered fragments.

Relevance Reasoning (Levy et al., 1997)

Task Simulation, Explanation.

Primitives Model fragments organised in assumption classes. The fragment library carries extra information in the form of modelling constraints, which indicate relevance relations among objects and the context in which they can be used.

Selection Process Models are composed in a "bottom-up" fashion: First the assumption classes that must be present are determined. Then the simplest fragment of each class is selected. Using the relevance reasoning modelling constraints, the initial selection of model fragments is augmented to include all relevant entities, while checking that the assumptions which have to be introduced are valid with the existing fragments assumptions. Fragments are also included if they help determine the non-exogenous variables appearing in the equations of already selected fragments.

Additional Refinement Refinements include checking for the validity of the aggregation of assumptions of the selected fragments, but also checking for the relevance and significance of the fragments selected with respect to the correctness of the simulated behaviour.

Bayesian Network Fragment Selection This is the content of this thesis but has also been outlined in (Biris and Shen, 1999).

Task Explanation.

Primitives Model fragments organised in assumption classes, in a component-oriented way: each assumption class focuses on a specific component. The fragments may be elaborated to carry extra information regarding the different behaviours exhibited by the corresponding component, resulting in more compact representations of the details associated with a component's structure and behaviour. Assumption classes are not restricted to have fragments with mutually contradictory conditions.

Selection Process Models are composed in the following fashion: First a Bayesian network of the associated fragments is constructed following the method outlined in (Biris and Shen, 1999), as well as in Section 4.2. Using the decision of an Approximate User Monitor that maps fragment detail to user expertise, appropriately detailed fragments are selected for those components that appear in the user query. The Bayesian network inference mechanism determines the fragments to use for the remaining components.

Additional Refinement None. The framework assumes that equation checking is done separately during the design stage of the model fragments, prior to any usage of the model fragments for explanation generation. However, the usage of Approximate Reasoning methodologies allows for the handling of uncertainty related to user requirements.

An initial vector of attributes that can help with the comparison follows below. The listed items cover the information setup, the actual reasoning process and the outcome of the fragment selection. All comparisons were made using the key features indicated in the available literature, in a qualitative fashion i.e. without complete statistical feature comparison in an experimental environment, as there was inability to retrieve actual details from the authors of the systems or from personal experience (again, due to lack of complete software implementations of these two approaches).

1. Typical complexity of fragment library used, in terms of how many fragments are usually considered in the process of selecting the best representation for domain system components and their associated phenomena. Due to lack of resources, as explained previously, this measure could not be extended to include an average of the resolution levels in the fragment representations contained within the libraries used for specific domain systems, though such a statistical analysis would be more desirable.

2. Whether or not any refinement steps are required to validate the formulated model.

3. Relative to the explanation-task, is the user expertise exploited in any way to guide the model fragment selection process? If so when does this happen (e.g. initial step, refinement steps) and how is it implemented? Does the implementation address at all any issues surrounding the uncertainty that accompanies the task of explanation generation based on interaction with users?

Representation complexity and immediacy of the selection process are of the greatest importance for the comparison (i.e. the first two items in the list above). The exploitation of user expertise is an important aspect for the framework of Bayesian network fragment selection since it forms part of a larger system that facilitates the generation of explanations for user training.

4.6.1 Comparison to CM Based on Causal Approximations

Nayak's position regarding the selection mechanism for model fragments originates in the proof that the selection process is indeed intractable (Nayak, 1994). Thus, certain measures are taken to ensure that the complexity of the selection algorithm is reduced from NP to P. The work addresses both of the aspects of the model formulation task, namely model assembly and model formulation, with the goal of alleviating some of the total algorithmic complexity involved.

The attempt against the fragment selection intractability led Nayak's work to the exploitation of a mechanism for following causal influences between model fragments, representing structural or behavioural aspects of the components for a given device. By following these causal links it is possible to determine from an initial set of fragments, selected in order to represent the key components in the given description of the device's expected behaviour, which other fragments should also be selected to complete the device model. These causal influences have to be determined explicitly via the *component interaction heuristic*, which essentially follows representational descriptions of the connections between the components involved, as specified in the fragments library.

For the Bayesian fragment selection technique, there is a similar determination of structural associations of fragments. However, the selection of relevant fragments occurs as a natural product of the Bayesian network reasoning process. The component interaction heuristic is replaced by the exploitation of structure, prior probabilities and inference within the Bayesian network. Both methods exhibit similar complexity and expressiveness in the knowledge representation associated with the model fragments library and with the underlying reasoning methodology.

Having identified the assumption classes that should be provided with fragments for the final model, Nayak's algorithm first selects the most complicated fragment from each class to create the initial device model. In contrast, the Bayesian network attempts to select the most appropriately detailed fragments right from the start of the model formulation process. In cases where the fragments for a component vary even moderately in their detail levels, the Bayesian network fragment selection can lead to substantial savings in formulation time. Admittedly, however, providing an optimal configuration regarding the network structure and prior probabilities may require some experimentation. Nevertheless, since the usage of the framework which employs the Bayesian fragment selection process is mainly for explanation purposes, it is possible to overcome some of the difficulties regarding the configuration of the Bayesian network via pre-specifying the networks that aim to represent the domain systems used for a training session.

Having determined the initial model for the domain device that is being modelled, Nayak employs a model simplification stage in order to reach an adequate model, i.e. a model whose representation is simpler (and thus easier to store, and reason with), whilst remaining adequate enough to fulfil the purpose of its creation such as providing explanations to users. This may take several reasoning cycles of Nayak's algorithm, since it needs to be applied to all the fragments involved, verifying each simplification in order to guarantee that the model is still adequate for usage.

In the Bayesian network fragment selection there is no such process of refinement of the fragment selection result after the network completes its reasoning. It is assumed that the equations and conditions associated with the individual model fragments are separately validated prior to the selection process, preferably during the design stage of the fragments. A minimum of actions can be taken by the fragment selection implementation module toward making any adjustments to the selection decision if the Bayesian network is undecided about selecting a fragment for a component, rather than altering all fragments appearing in a model. Thus, even though the Bayesian network selection process guarantees that a model fragment for each component in the examined device will be included in the formulated model, it cannot guarantee that the model is either accurate or correct mathematically since that depends on the input model fragments. However, the employment of approximate reasoning techniques to handle the indicated user preferences during explanation interaction sessions provides some measure of determination of representational detail in the selected fragments that matches the user requirements.

The refinement process that Nayak's method employs, eventually leads to a model that is both complete (contains all necessary fragments to explain the expected behaviour of the device at hand) and adequate (employs only the necessary complexity level with respect to describing the actual domain phenomena associated with the components of the particular device). In this aspect the model provides only the information necessary for explaining the expected device behaviour, avoiding unnecessary complexity. Therefore the final model can be deemed as being intuitive with respect to the particular level of complexity chosen to be the target of the refinement procedure.

However, the methodology followed by Nayak for the entire model composition process does not utilise, although it has the capability of doing so, any information regarding the user expertise or any user requirements toward selecting fragments of appropriate complexity. In contrast, the Bayesian fragment selection process is designed based on the principle of utilising all available knowledge for the user requirements in order to pre-select fragments that are associated with these requirements. This initial selection drives the consequent decision making process of the Bayesian network, and thus improves the intuitiveness of the finally formulated model with respect to the user expertise.

4.6.2 Comparison to CM Based on Relevance Reasoning

In the work of (Levy et al., 1997) the initial view taken for the task of model formulation is that it can be considered as a general case of reasoning about the relevance of knowledge that can be utilised to create a model. The task for which the models are needed is to simulate particular devices, in order to determine characteristics of the time-dependent behaviour of these devices, particularly for prediction purposes. Intuitively, predicting how values change over time requires knowledge of what type of causal influence is relevant to these changes.

To facilitate the subsequent reasoning on the relevance of all the possibly associated phenomena to a device behaviour, the authors enriched the representation of model fragments, by introducing constructs that enable declaration of the relationships between model fragments in a given library. The fragments are first grouped together under composite fragments, which populate the assumption classes of the model fragments library. The composition of fragments is effectively a combination of operating modes described by different fragments, similar to the combination of operating modes that occurs with fragments which are targeted for generating explanations given to experts of a particular domain in the Bayesian fragment selection case. Whereas in the Bayesian fragment selection framework this combination is necessary in order to encapsulate all knowledge that such expert users may require from a component fragment, in Levy's work it is merely a convenience scheme to help further group fragments within assumption classes. Thus, in the Bayesian fragment selection work, the composite fragments are not singleton sets ; only one fragment can eventually appear in the final model.

Further than combining fragments into composite model fragments, Levy et al also attach on the model fragments certain special conditions, in addition to the usual operating conditions. These are used as *meta-level* instructions, attempting to determine how relevant other fragments are to the representations of the phenomena provided by each fragment that is currently considered for selection. For instance, a description of a battery that ignores thermal aspects may be based on the claim that the temperature property is *irrelevant* to the query given at the system input. Another class of relevance claims consists of assumptions regarding the problem solving task, such as the presumption about temporal granularity that the final model should exhibit, or presumptions about the description accuracy to be used in that model.

There are no such explicit relevance assumptions in the representation used by the Bayesian fragment selection framework, at least not at the level used by Levy et al (i.e. determining relevance on the basis of which properties or variables are mostly relevant, or not, to a fragment's descriptions). The Bayesian network links provide a level of description of relevance relationships by associating together fragments of adjacent components in the device being modelled. Such relevance claims are only at the component level without detailing on whether certain parameters could be relevant or not for a particular fragment. The usage of such relevance claims, though they can help driving the search for selecting only the relevant phenomena associated with a given user query may not eventually improve the expressiveness of the fragments representation since they are local to particular component fragments. On the contrary it may explode the level of complexity in determining the consequences of preferring one fragment to another.

Having set the representation principles, using a complex web of modelling assumptions, as briefly described above, Levy et al exploit these assumptions to drive the selection mechanism. The selection process determines first which phenomena (and in particular which other quantities) are relevant to a query quantity, and subsequently determines which level of detail to use for modelling these phenomena. The process evolves by selecting the simplest fragments from all assumption classes that describe the query-associated quantities, and by performing a subsequent backward chaining to include other phenomena that are relevant to the initially selected fragments. The Bayesian network method contrasts this selection process, by attempting to select fragments that are associated with the user query *and* the indicated user preferences.

One final note has to do with the issue of utilising user expertise requirements to drive the selection or rejection of possible fragments. This is not supported directly by Levy et al, although it could be supported using relevance assumptions directly in the fragments representation.

4.6.3 Conclusion to the Comparison

Overall the Bayesian fragment selection seems to offer several attractive alternatives to traditional Compositional Modelling techniques, such as a similarly intuitive representation of model fragments, a graphical representation of possible causal associations between adjacent fragments with respect to the selection process, a coherent well established mechanism for propagating the effects of selecting an initial set of fragments, and finally a way to exploit indications regarding the user requirements for domain knowledge representation to drive the selection of those initial fragments.

However, it lacks several important characteristics and poses some points for future enhancements. The traditional methods for Compositional Modelling attempt to *discover* the associated phenomena when a user inputs a query about the domain visited. Since the Bayesian selection method uses only pre-constructed networks, this discovery becomes part of the network fabrication. Thus it is not an entirely automatic process, inherent of the model construction mechanism. Furthermore, whereas traditional CM provides extensive support for the validation of the final models, the techniques analysed in this thesis do not focus on this aspect, assuming that external processes will be provided to cover any verification of the input model fragments.

Chapter 5

Scalability and Complexity Issues

Manipulation and observation experiments
are what most people regard as proper experiments,
while exploratory and assessment studies seem informal,
aimless, heuristic, and hopeful. Testing hypotheses has
the panache of "real science", whereas exploration seems
like fishing and assessment seems plain dull. In fact,
these activities are complementary; one is not more scientific
than another; a research project will involve them all.

From P. Cohen's "Empirical Methods for Artificial Intelligence"

This chapter describes the issues regarding the scalability of the framework as it has been outlined so far. Intuitively, the described framework can provide the means to facilitate the selection of model fragments where this is a complex task due to the large number of possible selection alternatives. As such, it would prove most useful within a domain that can provide a great number of components each of which could be modelled using a large number of model fragment representations. In such a domain, approaches that attempt to elicit the library of fragments from first principles using a standard logic, as many of the mainstream CM techniques do, could prove too complex to complete successfully within tight time limits. The framework presented herein, attempts to facilitate the rapid selection processing of the available model fragments to provide some suggestions. In this aspect any elaboration of the scalability issues regarding the framework would prove (or disprove) its usefulness for such problem domains. Even if the fragments library does not really become very large, the study of the issues related to the scalability properties of the algorithms used in the framework would provide some intuition regarding the expected behaviour from the selection algorithm when the library size varies.

The main question that such a study attempts to answer is then:

> Once the size of the engineering system that is to be modelled for explanation generation becomes ever larger, what practical gains can the usage of Bayesian networks bring about to the task of selecting model fragments for the representation of those systems to interested users?

This is the question that motivates the present evaluation experiment. The goal is to provide evidence regarding the 'gains' in the question. The overall strategy for the experiment is thus: assuming that the variable is the size N of the library of model fragments then how do the run time and memory usage of the Bayesian network inference algorithm vary? To answer this experiments are designed to run in batch trials using artificially generated fragments and collections of fragments into assumption classes (representing domain system components). Note that the experiments do not test the effect that the size of the value set for each variable in the Bayesian network can have on the overall performance of the probability propagation algorithm. It has been shown that enumerating through all the value assignments for each network variable can be accomplished in linear time depending on the actual size of the value set, and thus it does not theoretically add to the complexity of the inference mechanism as significantly as other factors (Cooper, 1990; Dechter and El Fattah, 2001).

In order to be able to explain the results and identify possible sources of the issues in the efficiency of the algorithm used, it is necessary to provide a summary of the characteristics of the algorithm involved and its implementation within this particular project. This follows in the next section. The theoretical aspects of the inference mechanism used are outlined with respect to their effect on overall efficiency. Then, a description of the exploratory experimental procedure for the collection of results follows, in an effort to confirm how the theoretical expectations may hold for the probability propagation algorithm. Collected results are presented next, and rationalised, making several generalisations that follow the experimental results and providing indicators for possible future work.

5.1 Characterisation of the Shenoy-Shafer Inference Mechanism

The method used for the Bayesian Network inference mechanism, to drive the selection of model fragments, is a tree-clustering algorithm. Tree-clustering involves transforming the original problem specification into a tree-like representation form that can then be solved by a specialised tree-solving algorithm. Given a model formulation Bayesian network, the transforming mechanism identifies subproblems — groups of nodes in the Bayesian network — that together form a tree, and the solutions to these subproblems serve as the new probability distributions that can be associated with the corresponding groups of nodes. This tree-like representation that the Bayesian network is transformed to is also known as the *join-tree*. The following definitions provide some required terminology for the subsequent discussion.

Definition 5.1.1 (Moral Graph) *The moral graph of a Bayesian network is an undirected graph generated by connecting the tails of any two head-to-head pointing arcs in the network and removing the arrows.*

An example of a Bayesian network and its moral graph are shown in Figure 5.1.

Definition 5.1.2 (Ordered Graph, Ordered Graph Node Width, Ordering Width) *An ordered graph is a pair $(G;d)$ where G is an undirected graph and $d = X_1 \ldots X_n$ is an ordering of the nodes. The width of a node in an ordered graph is the number of its earlier neighbours. The width $w(d)$ of an ordering d, is the maximum width over all nodes for the given ordering.*

Definition 5.1.3 (Induced Width, Induced Graph, Triangulated Induced Graph) *The induced width of an ordered graph, $w^*(d)$, is the width of the induced ordered graph which is obtained by processing the nodes recursively, from the last to the first;*

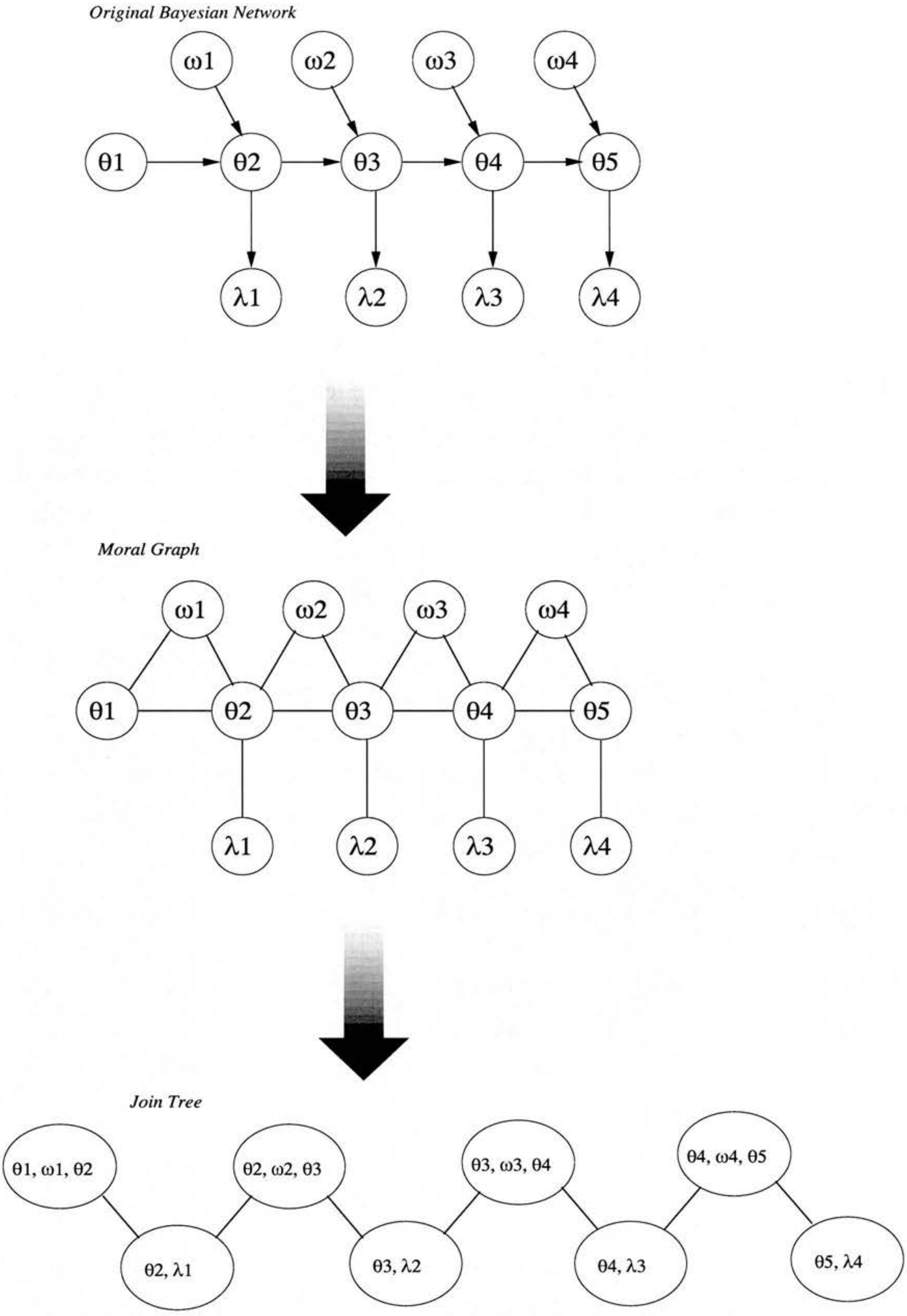


Figure 5.1: From Bayesian network to moral graph to join tree.

when node X is processed, all its earlier neighbours are connected. This process is also known as triangulation. After this process, the triangulated induced graph is chordal (i.e. every cycle of at least 4 nodes has a chord). The induced width of a graph, w^*, is the minimal induced width over all its orderings.

Definition 5.1.4 (Join-Tree Separator Size) *The separator size of a join-tree is the maximal size of the intersections between any two cliques, and the separator size s of a graph is the minimal separator size over all the graph's join-trees.*

Referring to the example given in Figure 5.1, the width of the moral graph for the ordering $d = \theta_1, \omega_1, \theta_2, \lambda_1, \omega_2, \theta_3, \lambda_2, \omega_3, \theta_4, \lambda_3, \omega_4, \theta_5, \lambda_4$ is 2. Since the moral graph is already triangulated then the induced width w^* is also 2. The separator size for the join tree is 1. Note that, in general, it is always true that for a join tree of separator size s and induced width w^*, $s \leq w^*$.

It has been shown in (Dechter and El Fattah, 2001), that the task of discovering posterior probability distributions using a tree-clustering algorithm is a process that exhibits time and space exponential scalability, depending on the size of the maximal clique, namely, it is exponential in the moral graph's induced width. In fact, the evidence Deter and El Fattah provided suggest that at best the space complexity can be expressed as exponential on the separator size of the join tree, which, given the fact that $s \leq w^*$, is theoretically an improvement. However, for this research because $s \equiv w^*$ it remains that the algorithm is expected to behave as $O(n \cdot \exp(w^*))$.

For the present work the meaning of this theoretical expectation is that as the number of fragments used for each component increases the performance of the selection algorithm will deteriorate rapidly, as will the storage requirements for data structures that are used during the run of the probability propagation algorithm. This is because by increasing the size of each assumption class, the induced width for the resulting tree at the neighbourhood of each inflated assumption class will increase, causing the rapid deterioration of the probability propagation within that neighbourhood and affecting global performance.

On the other hand, if the number of fragments used to describe each component is kept static, then the addition of new components will also cause deterioration of performance. This time, it is due to the increase of the overall propagation length for the join tree, but the deterioration will not be as rapid as when increasing the number of fragments within the assumption classes. In fact, the increase of the length of the join tree is expected to be a source of polynomial time complexity for the propagation algorithm. Despite the availability of theoretical results that mandate these complexity indications there is little work done on empirical assessment of the particular propagation algorithm. This chapter covers some of the missing links, regarding the exploration of the algorithmic execution during the propagation of probabilities around the network.

The fact that the overall propagation performance has a vulnerable dependence on the induced width of the underlying join tree prompts the question whether or not it is possible to create a join tree with a minimal induced width. Furthermore, due to the possibility of having many triangulation alternatives during the construction of the join tree, it is possible to have more than one join tree derived from a single Bayesian network. The algorithm used for the construction of the join tree attempts to populate join tree nodes with variables by always selecting those that have the minimum number of neighbours. In so doing, it is guaranteed that, at least the join tree used has a minimum induced width, compared to all possible join trees that can be constructed for the same Bayesian network, based on the Definitions 5.1.2 and 5.1.3.

Finally, it should be mentioned that once the propagation algorithm has terminated the implementation of the selection process involves searching within the vector of the fragments to find all the fragments which have scored the highest probability for selection. This is implemented using a Standard Template Library (STL) C++ vector structure, which guarantees that the process of processing each vector element in turn is at worst linear with the size of the vector. For this reason, the study that follows will focus on the propagation algorithm evaluation, considering the post-propagation step of selecting the winner fragments as non-essential to the overall complexity.

5.2 Scalability Assessment Procedure

The model fragment selection algorithm requires as inputs a set of model fragments, a Bayesian network initialised with prior probabilities and evidence regarding the selection of an initial subset of the given model fragments. The procedure to follow in order to measure the algorithm's run times is provided below:

---

1. Generate a sample of a Bayesian network of model fragments given certain criteria for its structure and the required information contents of the fragments.
2. Generate a sample of incoming evidence for certain fragments within this network. The amount of evidence provided is kept constant for simplicity.
3. Run the model fragment selection process to propagate the evidence in the network and select one fragment from each assumption class, depending on the posterior probabilities distribution.

Note that as the main concern is the assessment of run-time performance, the correctness or validity of the artificially generated model fragments in step 1 above is not of essence. The generated fragments should however contain at least a number of variable definitions with associations that determine which variables act as inputs and which as outputs within each fragment, so as to allow the prior probability heuristics to work. With that in mind the fragments and the Bayesian network can be generated via a mechanism that uses:

1. The desired number of components for which fragments will be generated. This is the number of assumption classes in the library, and effectively the depth of the generated network.
2. A lower and an upper bound for the number of variables in the fragments within

each assumption class. This can be used to determine also the number of fragments within each assumption class as only resolution dimension is considered. In the simplest case, this assumes that each fragment in the class differs from its siblings just by one variable. From the resolution of each fragment a random number is generated to determine how many of the fragment variables will be regarded as outputs and the remaining variables are considered to be the fragment's inputs. Since these fragments are not meant for explanation generation there is no need for allocation of "internal" variables, i.e. variables that are not used as terminal inputs or outputs to the corresponding model fragment. By altering these bounds, the number of fragments in each assumption class can also be altered, and thus the width of the generated network can be manipulated locally for each assumption class.

Based on these the fragments are generated with a clear definition of their resolution, inputs and outputs. Additionally, a set of connections is generated to identify the interconnections between fragments, thus determining the Bayesian network. A random subset of the fragments created is selected to act as the provided evidence. By being able to regulate the depth and width of the generated network, it is possible to wrap the network creation in a batch process that feeds each of the networks in turn to the fragment selection algorithm. After creating the necessary internal structures that represent the network nodes and their interconnections, the selection algorithm propagates the selection evidence which is also provided. For each batch process, the run-time for the completion of the evidence propagation procedure is recorded, over a number of runs executed one after the other sequentially, in order to obtain an average of the run-times.

It should be noted that the experiments treated the worst case scenarios, where full connectivity was assumed between the alternative fragments of each assumption class. As such these results would provide upper limits for the monitored execution runtimes.

5.3 Results on Experimental Scalability

The experiments were executed using a DELL Inspiron i8100 laptop computer, with a 1.2 GHz processor and 512 MB of RAM, running RedHat Linux 7.2 (kernel 2.4.9-34). The Lisp code for `Pulcinella` was compiled using the CMUCL Lisp 18d compiler for Linux. Measured run times correspond to elapsed CPU time excluding time spent in low level operations such as garbage collection and system calls.

Figure 5.2 shows the run-times collected by varying the number of components, while keeping the number of fragments per component (i.e. per assumption class) constant. For this particular set of experiments the number of fragments per component was three.

Trying to fit these points to a curve, using a simple least squares method provides the fittings shown in Figure 5.3. As indicated in this illustration, the complexity curve that matches closest the collected measurements is approximately quadratic $O(a \cdot n^2)$, with the cubic also lying almost on top of the quadratic curve. In fact the mathematical formulae that express these two curves are (both these formulae were determined as part of the fit algorithm used by *Mathematica*, the mathematical package employed to perform the fittings):

- For the quadratic curve: $0.0115193 + 0.611647 \cdot x + 0.0053456 \cdot x^2$.
- For the cubic fit:
  $-0.239107 + 0.663875 \cdot x + 0.00299629 \cdot x^2 + 0.0000283783 \cdot x^3$.

Apparently the very small factor of the cubic power in the second formula causes the cubic curve to behave as a very close approximation to a quadratic, and that is the reason why the cubic curve is so close to the quadratic. Based on this, it is reasonable to assume that the quadratic is the curve that fits best the collected data.

This is indeed what the theoretical studies indicate, namely, provided that the width of the induced graph that corresponds to a Bayesian network is kept constant, the probability propagation algorithms are generally of polynomial complexity on the number of nodes that appear in the network. It is worth noting that, the maximum number of

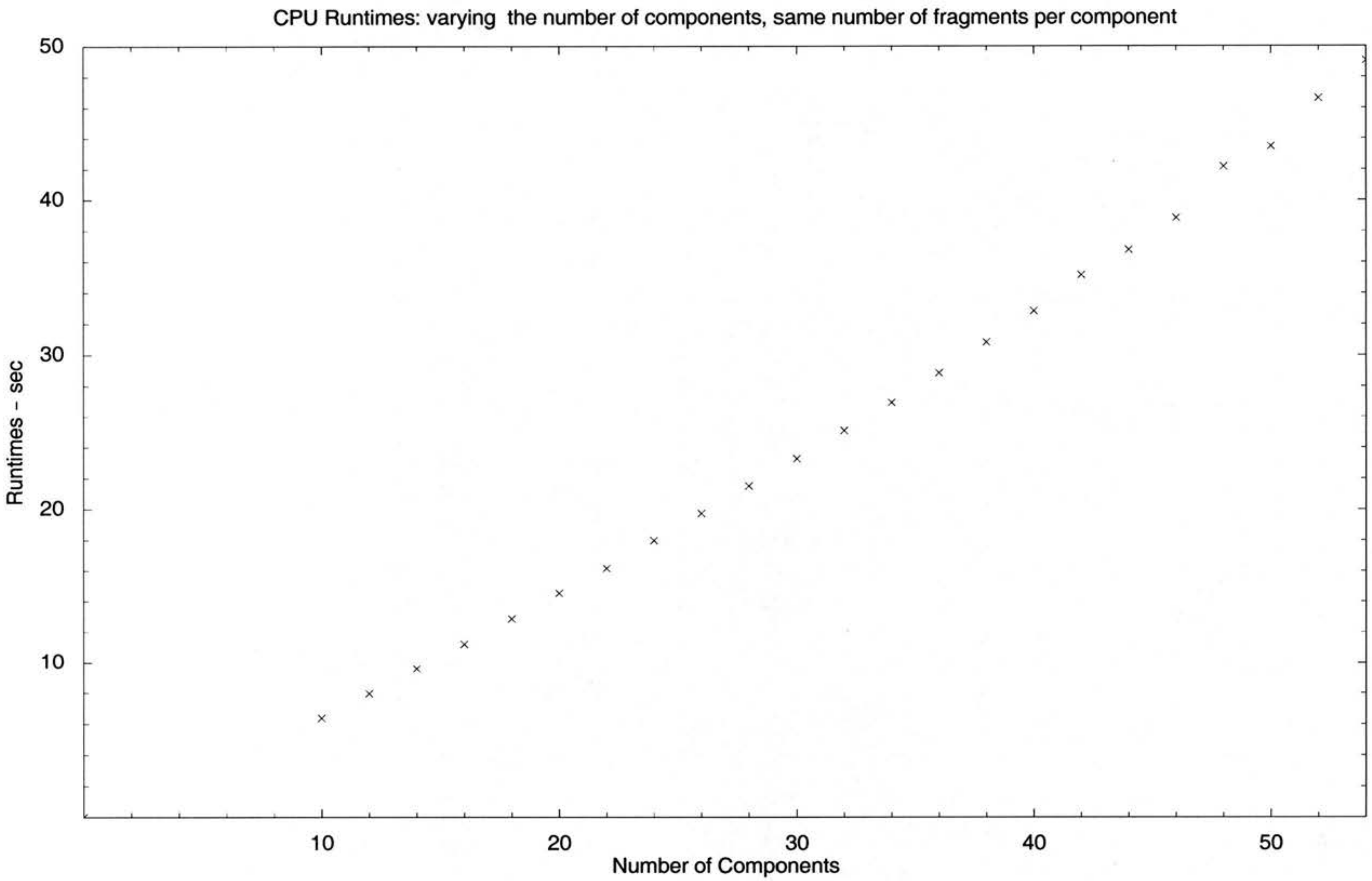


Figure 5.2: Scalability data: run-times when varying the depth of the tree.

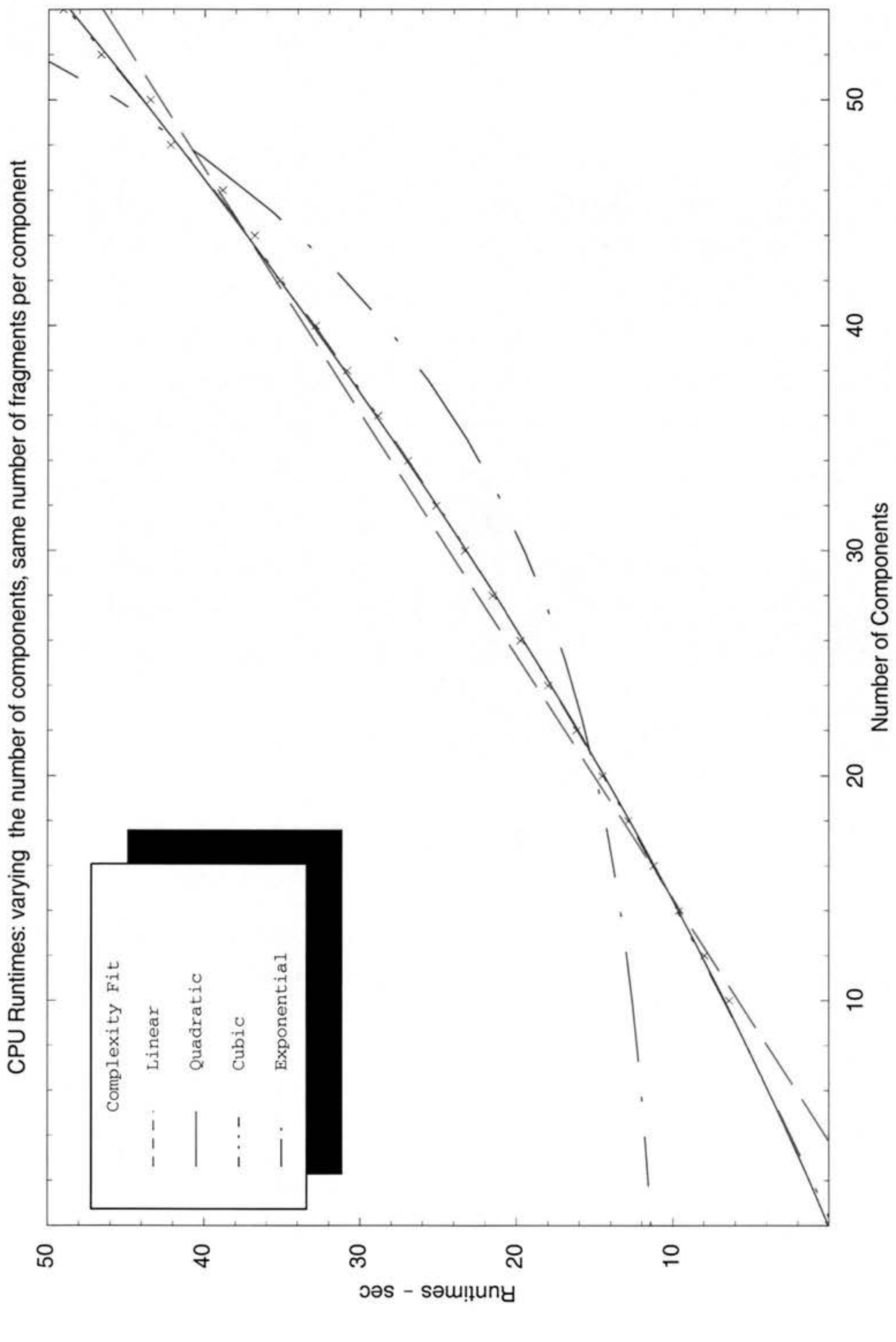


Figure 5.3: Scalability data: fitting data from Figure 5.2.

components that was possible to use for this set of experiments was not set by choice but by hard limits dictated by the platform used for the tests. Any tests using a set of components sized beyond this upper limit was terminated due to exceeding memory resources on the test platform.

The same compliance to theoretical expectations can be observed by experimenting with variance of the Bayesian network induced width. For this experiment the total number of components remains constant, while the number of fragments within the available assumption classes varies to cover a range of induced width values. The measured value is again the CPU runtime, that was required for the propagation algorithm to reach equilibrium after the insertion of evidence. Once again, the curve is fit using least squares, and clearly the increase of the induced width of the Bayesian network increases the required runtime exponentially. Although the algorithm implementation attempts to optimise the propagation behaviour by structuring the join-tree in a manner so as to minimise the induced width of the tree, the upper limit of model fragments used in each assumption class for the particular platform was five. The execution failed to complete for any configuration with more fragments per assumption class than that.

5.4 A Typical Large Domain System

The initial impression from the acquired results is that the propagation algorithm is limited, since it can fail when a variation of the input fragments occurs that is not so impossible to encounter in reality.

In practise, however, the use of only three fragments to each component as described in Chapter 4, to represent information for novice, intermediate and expert users should help ensure that the complexity due to the resulting induced width is kept under control. Nevertheless, how realistic is this with regard to anything larger than the simple thermodynamics examples illustrated in Chapter 4? In this section a sample domain system that is larger than the examples provided in Chapter 4 is provided to examine the possibilities.

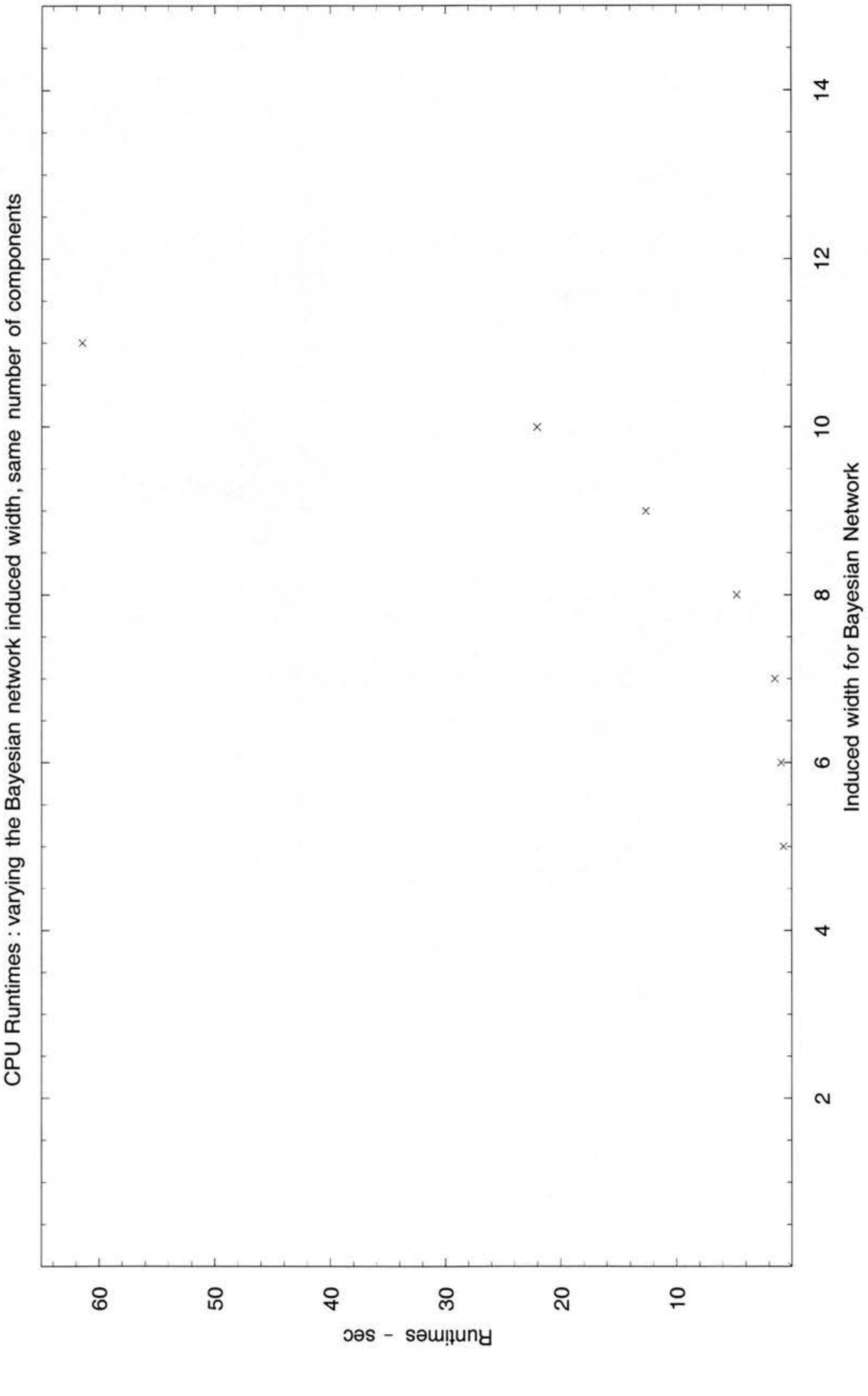


Figure 5.4: Scalability data: run-times when varying the network induced-width.

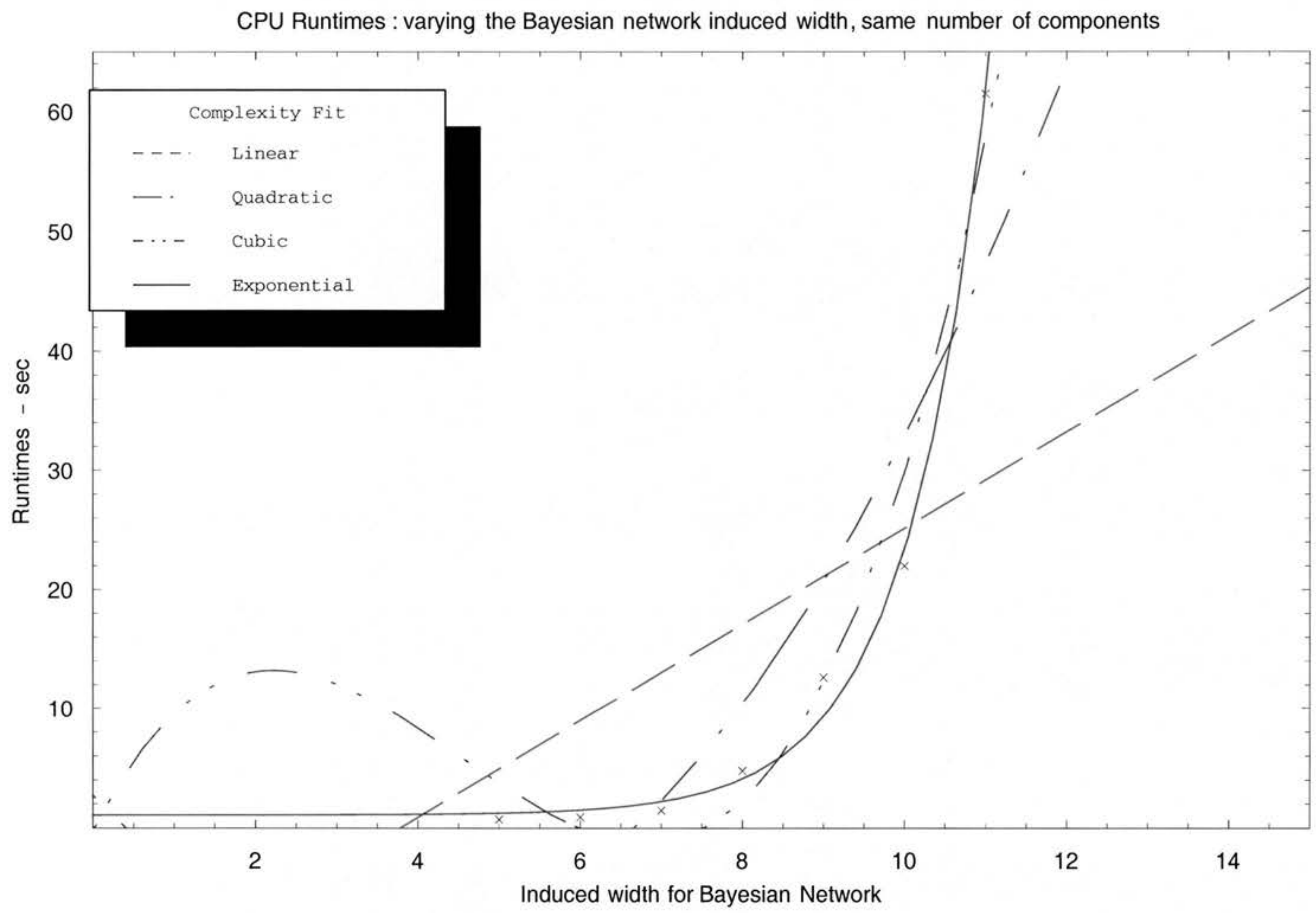


Figure 5.5: Scalability data: fitting data from Figure 5.4.

5.4.1 Domain System: a Robot

Figure 5.6 provides an outline of the main architecture of a small RugWarrior type of robot (Jones and Flynn, 1993), which is used for the present investigation. It is abstracted from the real schematics of the RugWarrior robot, which contains a greater variety of sensors and more components within the main-board. Despite the simplifications, the example includes all the essential components and it is a realistic possibility for the type of systems that can be handled by the framework. In particular, it consists of:

1. One bumper sensor that detects collisions, one infra-red sensor for detection of proximity to targets and one microphone for sound input,
2. A main-board component that carries the core CPU, connection ports to the sensors and actuators (motors) and the power source that is consumed by the components in the main-board and the motors. The unit contains a basic RAM component (stored in the CPU ROM) for storage of intermediate commands and parameter values during the execution program used to drive the robot .
3. Two motors and a motor interface that translates signals from the main-board into voltage variation for the motors, thus navigating the robot.

5.4.2 Bayesian Network for the Robot System

Based on the schematic of Figure 5.6, the following Bayesian network can be constructed, following the general principles outlined in Chapter 4. The final network is illustrated in Figure 5.7.

A join tree can be obtained by transforming the Bayesian network that has a maximum induced width of 14. The network results in a complete model fragment selection in 1200 seconds, in average, per run. This example system, though not containing assumption classes that involve more than three model fragments, it involves a high connectivity between different components, causing the final maximum of the induced

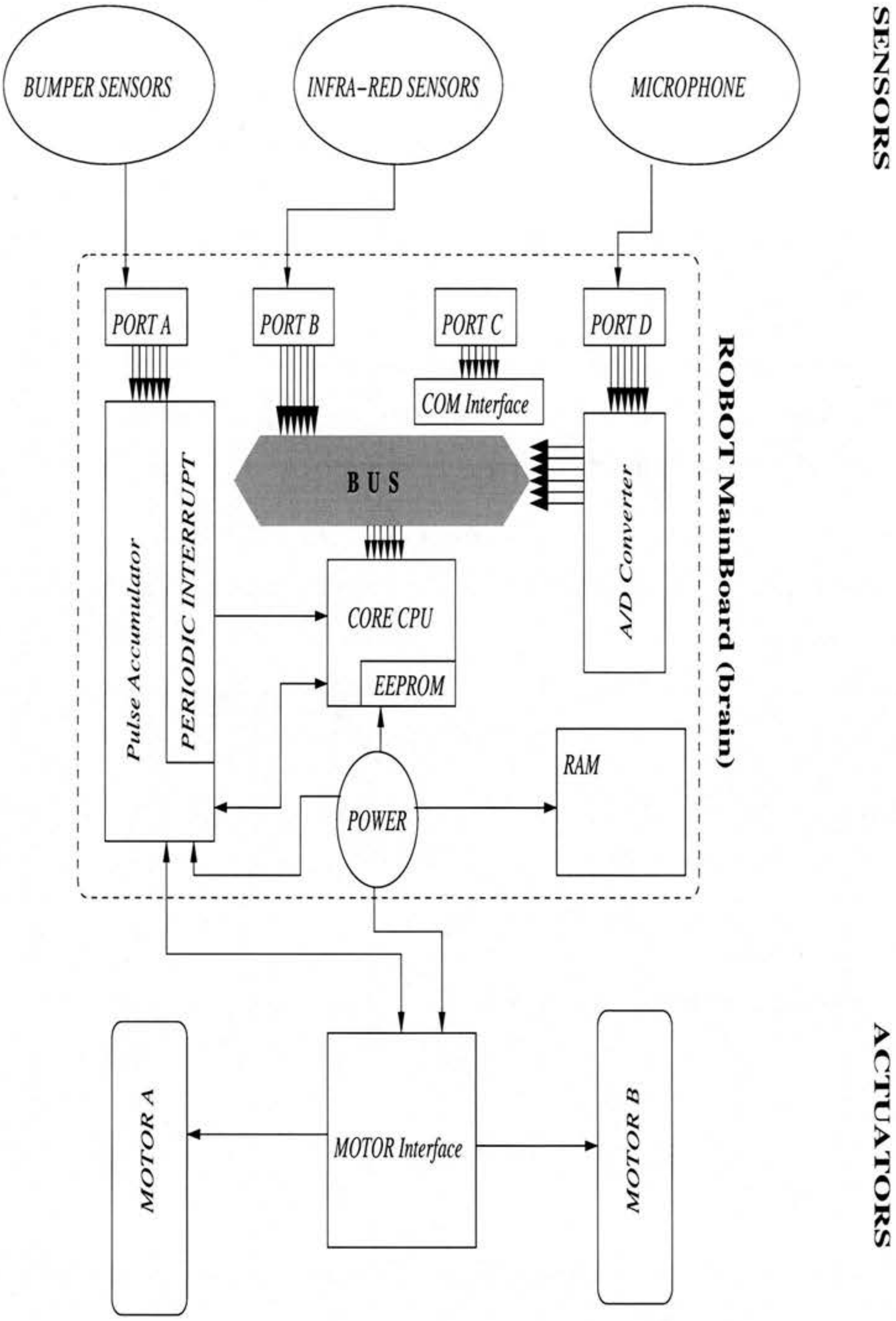


Figure 5.6: A RugWarrior type of robot: outline of main architecture.

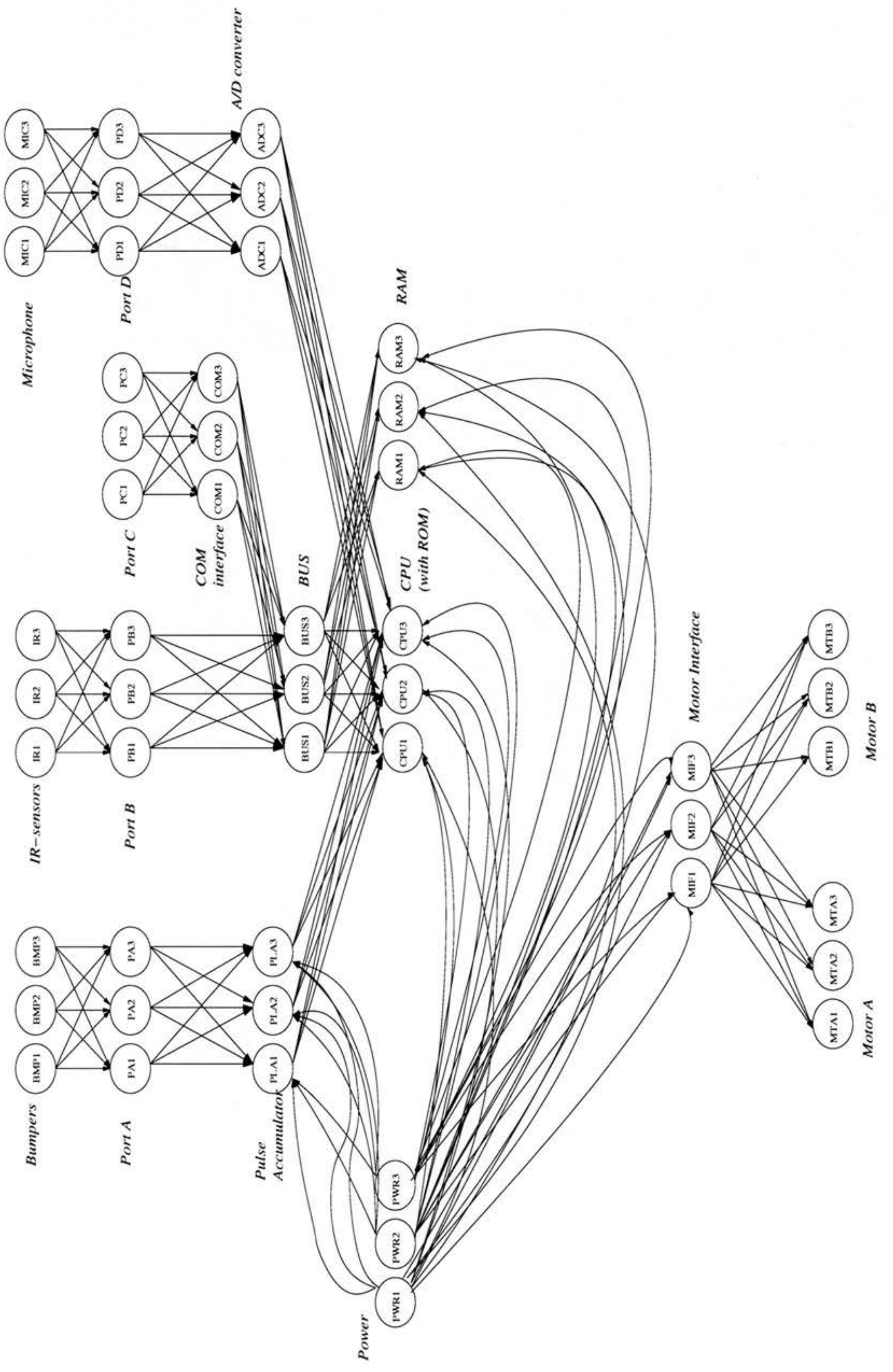


Figure 5.7: Robot system Bayesian network.

width parameter to reach critical levels for the performance of the propagation algorithm.

Despite being an example of a characteristic domain system that offers increased connectivity between the alternative model fragments, this system is actually typical of real-world systems. There are systems that may fail with the propagation algorithm and the implementation used, because of the inherent complexity of the algorithm depending on the graph induced width. Even in simple systems with just a few components, if the number of alternative fragments per component is increased, thus causing an increase of the join tree induced width, the algorithm would prove problematic, since its run-time complexity increases exponentially with the underlying join tree induced width.

5.5 Results and Suggestions

From the experimental results certain conclusions can be drawn and some recommendations can be suggested to the prospective user of the framework. These are summarised below.

Number of components This proved to be not so crucial for the performance of the evidence propagation algorithm. In fact it was shown empirically that, at worst, there is a quadratic type of complexity between the runtime and the number of components appearing in a target domain system. This is not so desirable as linear increase, but at the same time it is much better than exponential.

Number of model fragments in an assumption class This proved to be crucial to the overall performance of the fragment selection algorithm, due to the fact that it can increase the induced width of the join tree that corresponds to the original Bayesian network, thus making the entire selection procedure vulnerable to the exponential increase of runtime. As such, it should be born to mind that the assumption classes should be maintained to include a number of alternative fragments as low as possible.

Number of interconnections between model fragments This proved also to be associated with the potential increase of the induced width of the join tree that results from the original Bayesian network. Even if the assumption classes are kept to a low density of constituent model fragments the join tree could still demonstrate high connectivity because of the connectivity between components.

The selected propagation algorithm bears the marks of its vulnerability for performance. However it is quite general and simple to set up, and offers the ability to select various mediums (e.g. not only probability, but also boolean values, or possibility distributions) for the representation of the degrees of belief via the use of the more general concept of valuations. Its implementation through more efficient memory management would also push the complexity barriers further, thus increasing the capacity to enable the modelling of more complex domain systems with higher interconnectivity and more densely populated assumption classes. That is left to be proven in the future.

Chapter 6

Conclusion and Discussion

This thesis described a novel framework for model formulation, targeting primarily explanation generation as the consumer task for the formulated models. This chapter provides a summary of the results, an enumeration of the main contributions and a vision for future research.

6.1 Approximate Model Formulation for Explanation Generation

The framework described in this thesis provides a novel approach for model formulation, aiming for better handling of the uncertainty that accompanies this task when the primary usage of the formulated models is to generate explanations for users of a certain knowledge level. The model formulation is based on assembling together model fragments, which represent the structure and behaviour of parts of the domain system that is required for explanation. Knowing which model fragments to select, when there is evidence of preference for some of them, is considered paramount to being able to provide a set of fragments which comply with the user preferences. Through employing a combination of fuzzy evidence preparation, using the history of user selections in an interaction with the program and an approximate reasoning inference engine, with a Bayesian-based evidence propagation mechanism it is possible to handle the uncertainty sources in the model formulation process.

In implementing this framework the following major achievements were accomplished:

A Bayesian network based model fragment selection mechanism (Section 4.2) This is the core of the framework for the handling of uncertainty in selecting model fragments. There are two aspects in the construction of the network: the definition of its structure, representing the decision influence connections between the network nodes, and the elicitation of appropriate probabilities for its nodes, representing the perceived strengths of these connections.

The network structure is left upon human decision to configure approximately. In the experiments carried out in this project the networks were always constructed based on schematic information regarding the connectivity of the components present in the domain system under consideration. Though this process could be automated to some extent, the final decision for the way in which fragments from different assumption classes, representing domain system components, would be connected with each other is based on expert knowledge. This is realistic for the application problems at hand, given the explanation of the operation of engineering systems as the task.

On the other hand, the elicitation of probabilities for a Bayesian network can be completely automated, as has been demonstrated. A number of probability elicitation heuristics can be employed to allow uniform allocation of probabilities for the entire network, whilst accounting for the factors that will be reasonably considered for the determination of influence strengths among the connected network nodes. The factors considered by the heuristics aim to provide the information required to select fragments that are maximally compatible with the given evidence in terms of their detail, such as the resolution levels of the connected fragments and the influencing variables between connected fragments. The automatic probability allocation mechanism can be employed as a method for the initialisation of the network nodes with prior probability distributions. However, the final decision is again left with the designer of the Bayesian network, who is expected to inspect and verify the generated probabilities for the same reason mentioned above, In particular, a great deal of the knowledge associated with elicitation of probabilities for Bayesian networks is normally subjective, so the best that may be expected from automated probability elicitation methods is an initial guess that can be enhanced subsequently via human intervention.

Once prior probabilities are allocated to the network nodes, standard efficient propagation algorithms can be used for the task of updating the probabilities for all the network nodes, given selection evidence for some of the nodes. The result of this process is a set of model fragments that are maximally compatible with the evidence, in terms of their level of detail. As the evidence has been fabricated taking into account the user preferences throughout his/her interaction with the program, the selected fragments are synchronised with the user preferences for the detail explanations that will be generated.

A fuzzy reasoning evidence fabrication engine (Section 3.2.2) The Bayesian network itself can only select model fragments based on an initial indication of which fragments the user prefers, or of which fragments would be compliant with the target of the interaction with the user. The correlation of these soft, qualitative factors into crisp, quantitative measures that the network propagation algorithm can utilise is performed by the Fuzzy Reasoning module.

The work towards defining the functionality of this module involves expressing the fuzzy rules that can guide the "guessing" of the information needs of the user, whilst attempting to advance the user's knowledge regarding the domain system of interest. The rules are effectively laid out by the program designer, reflecting the educational target of the program. Being implemented in terms of Fuzzy Reasoning using customisable predefined rule templates the Approximate User Monitor module can become a tool towards more effective guidance of the Bayesian network, and thus towards that of the final formulated model.

An XML-based handling of explanation content and explanation deployment strategy (Sections 4.1 and 4.4) A further contribution towards building the Framework is the usage of XML as a) the means to facilitate and standardise the extraction of information from the final models, when preparing the content communicated back to the user, and b) the means to facilitate the description of the dialogue strategy for communication of the extracted content. This work involved abstracting the content requirements, used for the extraction of information from the model fragments selected for the final formulated models, in order to construct the explanations to be provided to the user. Furthermore, the dialogue structure has also been designed using XML, allowing the developer of the program to employ different explanation strategies via altering the dialogue structure as indicated in (Cawsey, 1991). As XML can be highly configurable, the employment of this technology for the Explanation Generation module allows for a suitable way to guide the behaviour of the program for the information needs potentially set by different users.

6.2 Future Work

The work described in this thesis is by no means complete, in addressing the ambitious goal of automated generation of explanatory text for the operation of a targeted engineering system. Though it attempts to cover the entirety of the Framework definition, it leaves certain questions unanswered. These questions may form a prime candidate for future continuation of this work. Below follows an account of work that could be done at a future stage:

Bayesian network structure The procedure for the structuring of the Bayesian network is at the moment not only heuristic but subject to individual expert opinion. One possible way to tackle this in the future would be by capturing information from schematic diagram input that describes the domain system, as well as from user preferences for the explanation ahead. Depending on what parts of the domain system the user wants to focus on, the network structure could be altered so that the assumption classes for these parts become the leaf nodes for the network (i.e. the evidence nodes). An interesting extension would be to use more modelling dimensions (Leitch et al., 1999) than just the model fragments' resolution, as was used in this thesis. Introducing more dimensions would require significant reworking of the semantics and processes around the structuring and inference of the Bayesian networks.

Dynamic alteration of the fragments' structure The framework provides general guidelines for the type of information that is required to exist in the available model fragments, for the subsequent information extraction to succeed. A more dynamic way of notifying the explanation generation modules about the information present in the selected fragments could be useful in allowing the program to determine useful content on the fly. In fact, it would be interesting to allow the dynamic alteration of the domain system of interest, either by allowing the user to interact and experiment on different possible system configurations, or by providing the ability to handle multiple similar systems in the same session.

Interaction through simulation In relation to the last issue, the current implementation of the Framework is static, in that it does not allow simulation of the generated models. Such a feature would be advantageous in a learning environment, provided that the user can also interact with the program experimenting with different configurations of the domain system(s) available for training. The ability to extract training information from simulation results would be a great advantage to any training system. Enabling explanation generation through simulation could also allow handling of other kinds of user queries, such as "how does component X relate to component Y", which can only be answered via a simulation of the interaction of the two components within the domain system function.

Approximate User Monitor The rules designed for use by the Approximate User Monitor are also subjective. A structured experiment probing the behaviour of real users with respect to explanation generation prototype programs, would allow the collection of more realistic rules to guide the function of the Monitor. Most recently, there has been work carried out in this direction. An adaptation of such research results may well help this task.

User Interface Currently, there is a missing link between the selection of objects for their explanation and the presentation of the final explanations, as there is no user interface facilitating either of these tasks. Presenting the explanation to the user in a meaningful educational manner, employing text and graphics would be beneficial in conveying the extracted information from the formulated models.

User evaluation of the framework The functionality of the model formulation framework has been based on the intuitive understanding of the problem, whilst employing the generic theoretical principles that have been laid out in the available literature. In particular, intuition was used for the design and implementation of the inference involved in the Approximate Reasoning Monitor, and in the methodology for generation of explanation content and dialogue structure, by the Content and Dialogue Managers respectively. In order for the techniques described herein to be readily available for deployment in an industrial context it would be important to evaluate the functionality of the entire framework, and of its individual modules, against the explanation requirements of end users. The explanation needs of the target users should be probed using techniques, ranging from questionnaires to full-scale ethnograhic and sociological evaluation of users, and following the principles analysed in (Forsythe, 1995; De Rossis et al., 1995). These techniques offer an initial standardisation of the evaluation of user needs, overcoming issues relating to the specificity of requirements of the questioned users as individuals, which may affect the generality of the generated explanation content. The evaluation results should then be used to validate the explanations that can be constructed by the framework, in terms of structure and content. The evaluation – validation cycle could be done in an incremental and iterative fashion, adjusting the functionality of the framework to meet the expectations of different target user groups. Such an evaluation would help calibrate the structure of the Approximate User Monitor fuzzy rules, which is described in Section 3.2.2, as discussed in a previous point in this list. The amended rules could then reflect user requirements via ameliorating the AUM perception of the user expertise and his/her needs for content detail when identifying evidence for the Bayesian network model formulation engine.

Framework application in an industrial context In the current implementation of the framework the entire cycle of the explanation generation is not fully coded, with most notable omission being the user interface. As such, the framework cannot be readily deployed in an industrial environment. To be able to complete the deployment of the framework in a real world training context, realising the potential of this research, several improvements are required. First, the implementation of the entire design as outlined in Figure 3.1 should be completed by providing a user interface. Then the functionality of the framework's modules should be evaluated, iteratively and incrementally, against the target users as indicated in the previous point of this list. Based on such evaluation, a number of modifications could be identified, related to the structure of the rules used by the Approximate User Monitor, the handling of XML employed by the Content Manager and Dialogue Manager modules and the behaviour of the Discourse History module to reflect the needs of the target user audience. A prototype, fully connected, version of the design of Figure 3.1 would be the result of this process, and could be used as a version for further improvements.

Other domains The selection of engineering domains offers a certain standardisation of the types of content provided by domain models. Targeting other domains, such as ecology, could prove to be challenging at least in the design of the fragments library, as well as the extraction of information for the generation of explanations. In validating the framework's operation by using domain systems from these domains more work could be done in areas such as the initial elicitation of representations for the model fragments and the expansion of the capabilities of the explanation generation to handle models from these domains.

Appendix A

Rules used in the Approximate User Monitor

As described in Section 3.2.2, the Approximate User Monitor obtains its input by evaluating information stored in what is called the "profiles" of the User Monitor. From the information stored in these profiles, for a given object $object_i$, the average detail level (resolution) $\bar{R}_{\text{object}_i}$ of its representations in past explanations can be computed, as well as its rate of change of the resolution level $\dot{R}_{\text{object}_i}$ in the most recent representation of the same object, as the explanation process continues. These two measures, together with the frequency of occurrence of an object within all the profiles, F_{object_i} and the perceived user expertise level for the particular object $UEXP_{\text{object}_i}$, are used by the Approximate User Monitor to compute the *desired resolution level*, $\hat{R}_{\text{object}_i}$, of the fragments to be selected in the subsequent model formulation process. $\bar{R}_{\text{object}_i}$ together with $\dot{R}_{\text{object}_i}$ provide some means to determine how the user has been interacting with the program so far to acquire explanations for $object_i$. To achieve a better view of the interest that he/she has shown in $object_i$ it is important to determine the frequency of requests for explanations for that object, via F_{object_i}. This measure provides an intuitive association of more frequent requests for information to a desire to find out more due to lack of understanding. $UEXP_{\text{object}_i}$, finally, provides some counterbalance to the possibility of misjudging an inquisitive expert for someone that may have trouble in perceiving the explanations given.

The following tables provide the setting for all the rule-sets used in the Approximate User Monitor, which is described in Section 3.2.2. The rules are not a simple Cartesian product of the input values, as there are some combinations of inputs that make no sense. In particular, the combinations:

1. $\dot{R} =$ Negative, $F =$ Zero and
2. $\dot{R} =$ Positive, $F =$ Zero

whilst varying the $\bar{R}$ and $UEXP$, are of no sense since the frequency value 'Zero' is a singular value (not a fuzzy set) signifying that a particular object has not been explained in previous interactions at all. When the frequency is 'Zero' there has been no change in the selected detail level for the representative model fragments for that domain object. The same holds for the combinations of $\bar{R} =$ Medium or High and $F =$ Zero: If something has not been explained before then there is no possibility for the average detail level used so far, $\bar{R}$, to be either Medium or High.

It is also worthwhile noticing that the rules do not attempt to cover the complete set of possible output value combinations. It is paramount though to have covered all the possibilities of input value combinations. All the rule-sets employed satisfy this constraint.

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
L	Z	Z	Novice	L	Novice	M	Novice	H	Novice	L	Novice
L	N	L	Novice								
L	Z	L	Novice								
L	P	L	Novice								
L	N	M	Novice	L	Novice	M	Interm	H	Interm	L	Interm
L	Z	M	Novice							L	Interm
L	P	M	Novice							M	Interm
L	N	H	Novice	L	Interm	M	Interm	H	Expert	M	Expert
L	Z	H	Novice								
L	P	H	Novice								

Table A.1: Fuzzy rule-set for $\bar{R} =$ Low and $UEXP =$ Novice

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
L	Z	Z	Interm	L	Interm	M	Interm	H	Interm	L	Interm
L	N	L	Interm								
L	Z	L	Interm								
L	P	L	Interm								
L	N	M	Interm	L	Interm	M	Interm	H	Expert	L	Interm
L	Z	M	Interm							L	Interm
L	P	M	Interm							M	Expert
L	N	H	Interm	L	Expert	M	Expert	H	Expert	L	Novice
L	Z	H	Interm							M	Interm
L	P	H	Interm							M	Expert

Table A.2: Fuzzy rule-set for $\bar{R} =$ Low and $UEXP =$ Intermediate

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
L	Z	Z	Expert	L	Expert	M	Expert	H	Expert	L	Expert
L	N	L	Expert								
L	Z	L	Expert								
L	P	L	Expert								
L	N	M	Expert	L	Expert	M	Expert	H	Expert	L	Expert
L	Z	M	Expert							M	Expert
L	P	M	Expert							M	Expert
L	N	H	Expert	L	Interm	M	Interm	H	Expert	M	Expert
L	Z	H	Expert	L	Expert	M	Expert				
L	P	H	Expert	L	Expert	M	Expert				

Table A.3: Fuzzy rule-set for $\bar{R} =$ Low and $UEXP =$ Expert

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
M	N	L	Novice	L	Novice	M	Novice	H	Novice	M	Novice
M	Z	L	Novice								
M	P	L	Novice								
M	N	M	Novice	L	Novice	M	Novice	H	Novice	L	Novice
M	Z	M	Novice			M	Novice	H	Interm	M	Novice
M	P	M	Novice			M	Interm	H	Interm	M	Interm
M	N	H	Novice	L	Novice	M	Novice	H	Interm	M	Novice
M	Z	H	Novice	L	Novice	M	Interm			M	Interm
M	P	H	Novice	L	Interm	M	Interm			M	Interm

Table A.4: Fuzzy rule-set for $\bar{R} =$ Medium and $UEXP =$ Novice

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
M	N	L	Interm	L	Interm	M	Interm	H	Interm	M	Interm
M	Z	L	Interm								
M	P	L	Interm								
M	N	M	Interm	L	Novice	M	Interm	H	Interm	M	Interm
M	Z	M	Interm	L	Interm	M	Interm	H	Expert	M	Interm
M	P	M	Interm	L	Interm	M	Expert	H	Expert	M	Expert
M	N	H	Interm	L	Novice	M	Interm	H	Interm	M	Interm
M	Z	H	Interm	L	Novice	M	Expert	H	Expert	H	Expert
M	P	H	Interm	L	Interm	M	Expert	H	Expert	H	Expert

Table A.5: Fuzzy rule-set for $\bar{R} =$ Medium and $UEXP =$ Intermediate

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
M	N	L	Expert	L	Expert	M	Expert	H	Expert	M	Expert
M	Z	L	Expert								
M	P	L	Expert								
M	N	M	Expert	L	Interm	M	Expert	H	Expert	M	Expert
M	Z	M	Expert	L	Expert						
M	P	M	Expert	L	Expert						
M	N	H	Expert	L	Interm	M	Interm	H	Interm	M	Interm
M	Z	H	Expert	L	Interm	M	Interm	H	Expert	H	Interm
M	P	H	Expert	L	Expert	M	Expert	H	Expert	H	Expert

Table A.6: Fuzzy rule-set for $\bar{R}$ = Medium and $UEXP$ = Expert

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
H	N	L	Novice	L	Novice	M	Novice	H	Novice	L	Interm
H	Z	L	Novice	L	Novice	M	Interm	H	Interm	H	Interm
H	P	L	Novice	L	Interm	M	Interm	H	Interm	H	Interm
H	N	M	Novice	L	Novice	M	Novice	H	Novice	M	Novice
H	Z	M	Novice	L	Interm	M	Interm	H	Interm	H	Interm
H	P	M	Novice	L	Interm	M	Interm	H	Interm	H	Interm
H	N	H	Novice	L	Novice	M	Novice	H	Interm	M	Novice
H	Z	H	Novice	L	Interm	M	Interm	H	Expert	H	Interm
H	P	H	Novice	L	Interm	M	Expert	H	Expert	H	Expert

Table A.7: Fuzzy rule-set for $\bar{R}$ = High and $UEXP$ = Novice

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
H	N	L	Interm	L	Interm	M	Interm	H	Interm	M	Interm
H	Z	L	Interm	L	Interm	M	Interm	H	Expert	M	Interm
H	P	L	Interm	L	Expert	M	Expert	H	Expert	H	Expert
H	N	M	Interm	L	Interm	M	Interm	H	Interm	M	Interm
H	Z	M	Interm	L	Expert	M	Expert	H	Expert	H	Expert
H	P	M	Interm	L	Expert	M	Expert	H	Expert	H	Expert
H	N	H	Interm	L	Novice	M	Novice	H	Interm	H	Novice
H	Z	H	Interm	L	Interm	M	Interm	H	Interm	H	Interm
H	P	H	Interm	L	Expert	M	Expert	H	Expert	H	Expert

Table A.8: Fuzzy rule-set for $\bar{R} =$ High and $UEXP =$ Intermediate

Inputs				*Complexity*							
				Low		*Medium*		*High*		*Unset*	
				Outputs							
$\bar{R}$	$\dot{R}$	F	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$	$\hat{R}$	$UEXP$
H	N	L	Expert	L	Interm	M	Interm	H	Expert	M	Interm
H	Z	L	Expert	L	Expert	M	Expert	H	Expert	M	Interm
H	P	L	Expert	L	Expert	M	Expert	H	Expert	H	Expert
H	N	M	Expert	L	Interm	M	Interm	H	Expert	M	Interm
H	Z	M	Expert	L	Expert	M	Expert	H	Expert	M	Expert
H	P	M	Expert	L	Expert	M	Expert	H	Expert	H	Expert
H	N	H	Expert	L	Interm	M	Interm	H	Interm	M	Interm
H	Z	H	Expert	L	Interm	M	Expert	H	Expert	H	Interm
H	P	H	Expert	L	Expert	M	Expert	H	Expert	H	Expert

Table A.9: Fuzzy rule-set for $\bar{R} =$ High and $UEXP =$ Expert

Bibliography

Addanki, S., Cremonini, R., and Penberthy, J. S. (1991). Graphs of models. *Artificial Intelligence*, 51:145–177.

Almond, R. (1995). Software for belief networks. World Wide Web. Online at http://bayes.stat.washington.edu/almond/belief.html.

Bhatnagar, R. (1994). Exploratory model building. In *Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence,UAI-94*, pages 77–85.

Biris, E. and Shen, Q. (1999). Automated modelling using bayesian networks for explanation generation. In *Proceedings of the 13th International Workshop on Qualitative Reasoning*, pages 19–26.

Bonissone, P. P., Henrion, M., Kanal, L. N., and F., L. J., editors (1991). *Uncertainty in Artificial Intelligence 6*. Elsevier Science Publishers.

Breese, J. S. and Fertig, K. W. (1991). Decision making with interval influence diagrams. In Bonissone et al. (1991).

Buchanan, B. and Shortliffe, E. (1984). *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison Wesley.

Carenini, G. and Moore, J. D. (1991). Generating explanations in context. In *Proceedings of the International Workshop on Intelligent User Interfaces*, pages 175–182.

Cawsey, A. (1991). Generating interactive explanations. In *Proceedings of the Ninth National Conference on Artificial Intelligence,AAAI-91*, pages 86–91.

Cawsey, A. (1993). *Explanation and Interaction*. MIT Press.

Cawsey, A. (1995). Developing an explanation component for a knowledge-based system: Discussion. *Expert Systems with Applications*, 8(4):527–531.

Clancey, W. J. and Letsinger, R. (1981). Neomycin: Reconfiguring a rule-based expert system for application to teaching. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 829–836.

Coletti, G. (1994). Coherent numerical and ordinal probabilistic assessments. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(12):1747–1754.

Cooper, G. F. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405.

D' Ambrosio, B. (1992). Real-time value-driven diagnosis. In *Proceedings of the Third International Workshop on the Principles of Diagnosis*, pages 86–95.

da Silva, F. S. C., Vasconcelos, W. W., Robertson, D. S., Brilhante, V., de Melo, A. C. V., Finger, M., and Agustí, J. (2002). On the insufficiency of ontologies: Problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems*, 15(3):147–167.

Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in bayesian belief networks is np-hard. *Artificial Intelligence*, 60:141–153.

Davis, P. K. and Hillestad, R. (1993). Families of models that cross levels of resolution: Issues for design, calibration and management. In *Proceedings of the Winter Simulation Conference*, pages 1003–1012.

De Rossis, F., Grasso, F., Berry, D. C., and Gillie, T. (1995). Mediating between hearer's and speaker's views in the generation of adaptive explanations. *Expert Systems with Applications*, 8(4):403–417.

Dechter, R. and El Fattah, Y. (2001). Topological parameters for time-space tradeoff. *Artificial Intelligence*, 125(1-2):93–118.

DeKleer, J. H. (1986). An assumption-based truth maintainance system. *Artificial Intelligence*, 28:127–162.

DeKleer, J. H. and Brown, J. S. (1984). A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83.

Driankov, D., Hellendoorn, H., and Reinfrank, M. (1996). *An introduction to Fuzzy Control*. Springer-Verlag.

Druzdzel, M. J. and van der Gaag, L. C. (1995). Elicitation of probabilities for belief networks: Combining qualitative and quantitative information. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence,UAI-95*, pages 141–148.

Falkenheiner, B. and Forbus, K. (1991). Compositional modeling: finding the right model for the job. *Artificial Intelligence*, 51:95–143.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence*, 24:85–168.

Forsythe, D. E. (1995). Using ethnography in the design of an explanation system. *Expert Systems with Applications*, 8(4):403–417.

Gallanti, M., Roncato, M., Stefanini, A., and Tornielli, G. (1989). A diagnostic algorithm based on models at different level of abstraction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1350–1355.

Geiger, D. and Pearl, J. (1988). On the logic of causal models. In *Proceedings of the 4th Workshop on Uncertainty in Artificial Intelligence*, pages 136–147.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.

Goldszmidt, M. and Pearl, J. (1992). A simple approach to belief revision, belief update and reasoning about evidence and actions. In *Proceedings of the 3rd International Joint Conference on Principles of Knowledge Representation and Reasoning*, pages 661–672.

Gruber, T. R. and Gautier, P. (1993). Machine-generated explanations of engineering models: A compositional modelling approach. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1502–1508.

Halliday, M. A. K. and Hasan, R. (1989). *Language, Context and Text: Aspects of Language in a Social-Semiotic Perspective*. Oxford University Press.

Henrion, M. (1986). Propagating uncertainty in bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence 2*, pages 149–163.

Henrion, M. (1991). Search-based methods to bound diagnostic probabilities in very large belief networks. In *Proceedings of the Seventh Annual Conference on Uncertainty in Artificial Intelligence,UAI-91*, pages 142–150.

Hobbs, J. R. (1985). Granularity. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1–4.

Hrycej, T. (1990). Gibbs sampling in bayesian networks. *Artificial Intelligence*, 46:351–363.

Hulme, M. (1995). Improved sampling for diagnostic reasoning in bayesian networks. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence,UAI-95*, pages 315–322.

Iwasaki, Y. and Simon, H. A. (1994). Causality and model abstraction. *Artificial Intelligence*, 67:143–194.

Jones, J. L. and Flynn, A. M. (1993). *Mobile Robots: Inspiration to Implementation*. A. K. Peters.

Kanazawa, K., Koller, D., and Russell, S. (1995). Stohastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence,UAI-95*.

Karnopp, D. C., Margolis, D. L., and Rosenberg, R. C. (1990). *System Dynamics: a Unified Approach.* John Wiley & Sons.

Keppens, J. (2002). *Compositional Ecological Modelling via Dynamic Constraint Satisfaction with Order-of-Magnitude Preferences.* PhD thesis, University of Edinburgh.

Keppens, J. and Shen, Q. (2001). On compositional modelling. *The Knowledge Engineering Review*, 16(2):157–200.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society, Series B*, 50(2):157–224.

Lee, C. C. (1990). Fuzzy logic in control systems : Fuzzy logic controller -parts i-ii. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2):404–418.

Lee, P. M. (1997). *Bayesian Statistics: An Introduction.* Arnold.

Leitch, R., Shen, Q., Coghill, G., and Chantler, M. (1999). Choosing the right model. *IEE Control Theory and Applications*, 146(5):435–449.

Lester, J. and Porter, B. (1997). Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. *Computational Linguistics*, 23:65–101.

Levy, A. Y., Iwasaki, Y., and Fikes, R. (1997). Automated model selection for simulation based on relevance reasoning. *Artificial Intelligence*, 96:351–394.

Liu, Z. Y. (1991). Tailoring tutorial explanations via model switching. In *Proceedings of the Contributed Sessions, 1991 Conference on Intelligent Computer-Aided Training (NASAConference Publication 10100, vol II)*, pages 383–403.

Liu, Z. Y. (1993). Qualitative reasoning about physical systems with multiple perspectives. *AI Magazine*, 14:77–79.

Liu, Z. Y. and Farley, A. M. (1991). Tasks, models, perspectives, dimensions. In *Proceedings of the 5th International Workshop on Qualitative Reasoning (QR '91)*, pages 1–9.

Luger, G. and Stubblefield, W. (1993). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* The Benjamin/Cummings Publishing Company Inc.

Mahoney, S. and Laskey, K. (1996). Network engineering for complex belief networks. In *Proceedings of the Twelveth Annual Conference on Uncertainty in Artificial Intelligence,UAI-96.*

McCalla, G. I., Greer, J. E., Bryce, B., and Pospisil, P. (1996). Granularity hierarchies. *Computers and Mathematics with Applications: Special Issue on Semantic Networks*, 23:363–376.

McKeown, K. R. (1985). *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press.

Miguel, I. (2001). *Dynamic Flexible Constraint Satisfaction and Its Application to AI Planning.* PhD thesis, University of Edinburgh.

Miguel, I. and Shen, Q. (1999). Hard, flexible and dynamic constraint satisfaction. *Knowledge Engineering Review*, 14(3):199–220.

Mittal, S. and Falkenhainer, B. (1990). Dynamic constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 25–32.

Mittal, V. O. and Paris, C. (1995). Generating explanations in context: The system perspective. *Expert Systems with Applications*, 8(4):491–503.

Moore, D. S. and McCabe, G. P. (2002). *Introduction to the Practice of Statistics.* W.H. Freeman.

Moore, J. D. (1995). *Participating in Explanatory Dialogues.* MIT Press.

Moran, M. and Shapiro, H. N. (1992). *Fundamentals of Engineering Thermodynamics.* John Wiley & Sons Inc.

Murthy, S. S. (1987). Qualitative reasoning at multiple resolutions. In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 296–300.

Murthy, S. S. and Addanki, S. (1987). Prompt: An innovative design tool. In *Proceedings of the 6th National Conference on Artificial Intelligence.*

Nayak, P. P. (1994). Causal approximations. *Artificial Intelligence*, 70:277–334.

Nayak, P. P. and Joskowicz, L. (1996). Efficient compositional modeling for generating causal explanations. *Artificial Intelligence*, 83:193–227.

Nayak, P. P., Joskowicz, L., and Addanki, S. (1992). Automated model genration using context-dependent behaviors. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 710–716.

Neches, R., Fikes, R. E., Finin, T., Gruber, T. R., Senator, T., and Swartout, W. R. (1991). Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56.

Paris, C. (1989). *User Models in Dialog Systems*, pages 200–232. Springer-Verlag.

Pearl, J. (1986). Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29:241–288.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Morgan-Kaufmann.

Pearl, J. (1993). Graphical models, causality and intervention. *Statistical Science*, 8(3):266–273.

Pearl, J. (1996). Structural and probabilistic causality. Technical Report R-237, UCLA Cognitive Systems Laboratory. Appearing in Shanks et al. (1996).

Pearl, J. (2000). *Causality*. Cambridge University Press.

Pearl, J. and Dechter, R. (1996). Identifying independencies in causal graphs with feedback. Technical Report R-243, UCLA Cognitive Systems Laboratory.

Pedrycz, W. and Gomide, F. (1998). *An Introduction to Fuzzy Sets: Analysis and Design*. MIT Press.

Poole, D. (1996). Probabilistic conflicts in a search algorithm for estimating posterior probabilities in bayesian networks. *Artificial Intelligence*, 88:69–100.

Pradhan, M., Henrion, M., Provan, G., del Favero, B., and Huang, K. (1996). The sensitivity of belief networks to imprecise probabilities: an empirical investigation. *Artificial Intelligence*, 85:363–397.

Raiman, O. (1991). Order of magnitude reasoning. *Artificial Intelligence*, 51:11–38.

Rassmussen, J. (1983). Skills, rules and knowledge; signals, signs, symbols and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3):257–266.

Ravindranathan, M. (1996). *Heterogeneous Intelligent Control Systems*. PhD thesis, Heriot-Watt University.

Ray, E. T. (2001). *Learning XML*. O'Reilly & Associates.

Reiter, E., Mellish, C., and Levine, J. (1995). Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9(3):259–287.

Rickel, J. and Porter, B. (1997). Automated modelling of complex systems for answering prediction questions. *Artificial Intelligence*, 93(1-2):201–260.

Saffiotti, A. and Umkehrer, E. (1991). Pulcinella: A general tool for propagating uncertainty in valuation networks. In *Proceedings of the 7th Conf. on Uncertainty in AI*, pages 323–331.

Schut, C. and Bredeweg, B. (1995). Supporting qualitative model construction: Eliminating incorrectly predicte derivatives. In *Proceedings of the 9th International Workshop on Qualitative Reasoning about Physical Systems*, pages 163–172.

Schut, C. and Bredeweg, B. (1996). An overview of approaches to qualitative model construction. *The Knowledge Engineering Review*, 11(1):1–25.

Shanks, D. R., Holyloak, K., and Medin, D. L. (1996). *The Psychology of Learning and Motivation, Vol. 34: Causal Learning*. Academic Press.

Shenoy, P. and Shafer, G. (1988). An axiomatic framework for bayesian and belief-function propagation. In *Proceedings of AAAI Workshop on Uncertainty in AI*, pages 307–314.

Shenoy, P. P. (1997). Binary join trees for computing marginals in the shennoy-shafer architecture. *International Journal of Approximate Reasoning*, 17(2-3):239–263.

Sime, J. A. (1994). *Model Switching in Intelligent Training Systems*. PhD thesis, Heriot-Watt University.

Sime, J. A. (1998). Model switching in a learning environment based on multiple models. *Interactive Learning Environments journal, "Special issue on the Use of Qualitative Reasoning Techniques in Interactive Learning Environments"*, 5:109–124.

Sinclair, J. M. and Coulthard, R. M. (1975). *Towards an Analysis of Discourse: The English used by Teachers and Pupils*. Oxford University Press.

Spiro, R. J., Feltovich, P. J., and Anderson, D. K. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. In *Proceedings of the 10th Annual Cognitive Science Conference*.

Struss, P. (1992). *Readings in Model-Based Diagnosis*, chapter What's in SD? Towards a Theory of Modeling for Diagnosis, pages 419–450. Morgan-Kaufmann.

Suthers, D. (1993). Preferences for model selection in explanation. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1208–1213.

Suthers, D. (1994). Strategies for sequencing as a planning task. In *Proceedings of the 7th International Workshop on Natural Language Generation*, pages 29–36.

Suthers, D. D. (1991). A task-appropriate hubrid architecture for explanation. *Computational Intelligence*, 7:315–332.

Swartout, W. R. (1983). Xplain: A system for creating and explaining expert consulting systems. *Artificial Intelligence*, 21(3):285–325.

Swartout, W. R., Paris, C. L., and Moore, J. D. (1991). Design for explainable expert systems. *IEEE Expert*, 6(3):58–64.

Verma, T. and Pearl, J. (1988). Causal networks: Semantics and expressiveness. In *Proceedings of the 4th Workshop on uncertainty in Artificial Intelligence*, pages 352–259.

Verma, T. S. and Pearl, J. (1990). Equivalence and synthesis of causal models. In Bonissone et al. (1991).

Weld, D. S. (1990). Approximation reformulations. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 407–412.

Weld, D. S. (1992). Reasoning about model accuracy. *Artificial Intelligence*, 56:225–300.

Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44:257–303.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons.

Wick, M. R., Dutta, P., Wineinger, T., and Conner, J. (1995). Reconstructive explanation: A case study in integral calculus. *Expert Systems with Applications*, 8(4):463–473.

Williams, M. D., Hollan, J. D., and Stevens, A. L. (1983). *Human Reasoning about a Simple Physical System*, chapter 7. Lawrence Erlbaum Associates.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8:338–353.