ELSEVIER Artificial Intelligence 96 (1997) 351-394

Artificial Intelligence

Automated model selection for simulation based on relevance reasoning

Alon Y. Levy[a,1], Yumi Iwasaki[b,*], Richard Fikes[c,2]

[a] *AT&T Bell Laboratories, 600 Mountain Ave., Room 2A-440, Murray Hill, NJ 07974, USA*

[b] *Knowledge Systems Laboratory, Stanford University, Gates Bldg. 2A, Rm. 256, Stanford, CA 94305, USA*

[c] *Knowledge Systems Laboratory, Stanford University, Gates Bldg. 2A, Rm. 246, Stanford, CA 94305, USA*

Received March 1996; revised August 1997

Abstract

Constructing an appropriate model is a crucial step in performing the reasoning required to successfully answer a query about the behavior of a physical situation. In the compositional modeling approach of Falkenhainer and Forbus (1991), a system is provided with a library of composable pieces of knowledge about the physical world called *model fragments*. The *model construction* problem involves selecting appropriate model fragments to describe the situation. Model construction can be considered either for static analysis of a single state or for simulation of dynamic behavior over a sequence of states. The latter is significantly more difficult than the former since one must select model fragments without knowing exactly what will happen in the future states.

The model construction problem in general can advantageously be formulated as a problem of reasoning about *relevance* of knowledge that is available to the system using a general framework for reasoning about relevance described by Levy (1993) and Levy and Sagiv (1993). In this paper, we present a model formulation procedure based on that framework for selecting model fragments efficiently for the case of simulation. For such an algorithm to be useful, the generated model must be adequate for answering the given query and, at the same time, as simple as possible. We define formally the concepts of adequacy and simplicity and show that the algorithm in fact generates an adequate and simplest model. © 1997 Published by Elsevier Science B.V.

Keywords: Model formulation; Relevance reasoning; Qualitative reasoning; Simulation

* Corresponding author. Email: iwasaki@ksl.stanford.edu.

[1] Email: levy@research.att.com.

[2] Email: fikes@ksl.stanford.edu.

1. Introduction

Models are the conceptual objects humans study instead of studying the real thing. Models themselves are products of our intellectual endeavor, and constructing an appropriate model for a problem is a challenging problem solving task in itself. Models are constructed for various reasons. For example, simplified models of the real world are constructed because the real world is too complex to comprehend in its entirety, and models of a device being designed are constructed in order to understand the behavior of the device by experimenting with the model even before the device is actually built.

There are as many possible models of a given subject of study as there are reasons for constructing models. There is no one "true" or "correct" model since any model is necessarily an abstraction and the goodness of a model depends on one's goal, i.e., the question one wishes to find an answer to by constructing and studying a model. For a model to be useful, it must contain enough information to answer the question with sufficient precision and accuracy without containing too much unnecessary detail. Constructing such a model requires deciding what information could be relevant for answering the questions and, therefore, should be included in the model.

Thus, model formulation can be considered as a special case of a more general problem of reasoning about relevance of knowledge for a given goal. Researchers such as Subramanian and Genesereth [30] and Levy [16] have proposed general frameworks for reasoning about relevance of knowledge. In this paper, we present an application of one of them, namely Levy's, to the problem of model formulation. We propose an efficient procedure, based on the general framework, for formulating a model for the purpose of simulation of physical devices. For any model formulation procedure to be useful, the generated model must be adequate for answering the given query and, at the same time, as simple as possible. In later sections, we will define formally the concepts of adequacy and simplicity and show that the procedure in fact generates an adequate and simplest model.

1.1. Motivations

The ability to analyze a physical system using a model of its behavior is an important skill required of engineers. Equally important, if not more, is the ability to formulate a model that is appropriate for one's purpose. However, the problem of how to build a good model is much less understood than that of how to analyze a model once it is formulated. Most computational tools intended to assist in analysis of model behavior rely on the user to construct a model.

The ability to formulate an appropriate model would enhance the utility of such systems greatly by making it much easier to take advantage of their analysis capabilities. For example, though simulation is a very useful tool for evaluating design alternatives in engineering design, it is not currently used as freely as it could be because of the cost involved in formulating a model, performing a simulation, and interpreting the results. If a system could quickly formulate an appropriate model for analyzing the particular aspect of interest, perform the simulation, and produce an interpretation of the results in a readily understandable form, a designer could much more easily analyze design

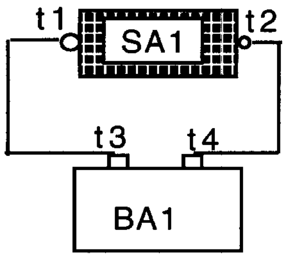


Fig. 1. An example circuit: SA1 is a solar array and BA1 is a rechargeable battery.

alternatives. Such a capability would enable designers to make better informed decisions during the design process, thereby improving the efficiency of the process as well as the quality of the final design.

1.1.1. Compositional modeling as a modeling paradigm

Compositional modeling [7] is an effective method for automatically formulating a behavior model represented by a system of ordinary differential equations for a physical system that can be adequately modeled as a lumped-parameter system.[3] In the compositional modeling approach, a system is provided with a library of composable pieces of knowledge about the physical world called *model fragments*. Each model fragment describes one aspect of a component behavior or a physical process. The system formulates a model of a given physical situation by selecting applicable model fragments and composing them.

The main advantage of compositional modeling is its modularity. Writing model fragments, each describing a single phenomenon, is a much easier task than composing a complete model for every possible system and query. Adding model fragments to an existing library is also much easier. Furthermore, model fragments can be reused in any appropriate context.

For a system intended to aid humans in analyzing a wide variety of behaviors of physical systems in a given domain, compositional modeling is a promising approach for automatically formulating a model. However, in order for the compositional modeling approach to succeed, one must have a mechanism for selecting model fragments in such a way that the resulting model will be appropriate for the given analysis problem. Also, it is imperative for the selection algorithm to have a reasonable time complexity, because all the savings achieved by using an appropriate model for analysis will not be worthwhile if the cost of the selection process itself is prohibitive.

1.1.2. Model formulation in compositional modeling

Selecting an appropriate model is crucial when the problem domain is rich with various levels of abstractions and different perspectives at which phenomena can be studied. Suppose one is analyzing the design of an electrical power supply consisting of

[3] When spatial variation is not of interest, one speaks of the system as being *lumped parameter* [6].

a rechargeable battery and a solar array as shown in Fig. 1. If one is interested in the variation in the voltage level supplied by the battery over the course of a day, one would construct a model that takes into account phenomena such as solar power generation and charging and discharging of the battery. Furthermore, the model would describe those phenomena with mostly electrical properties such as currents, voltages and charge level. If, instead, one is interested in the variation in the charging capacity of the battery over several months, one would have to take into consideration other phenomena such as aging, whose effect becomes observable only after many charge-discharge cycles. Yet another possibility is that one is interested in aging and in the details of the chemical processes that cause aging. In this case, an appropriate model would consist of representations of individual chemical processes involving the electrolyte and the electrodes of the battery.

While the library of model fragments may contain knowledge about a wide variety of physical phenomena, most of them may be irrelevant for any given problem. Each phenomenon can also be represented by several different model fragments, representing different ways to describe the same thing based on different assumptions about the representation and the problem to be solved, including such things as approximations made, desired temporal and quantitative accuracy, and precision.

Therefore, in the context of compositional modeling, the *model formulation* problem is:

- Select an appropriate subset of the model fragment library and appropriate instantiations of that subset,

given:

- A description of a problem situation, and
- An analysis goal,

A more formal statement of the problem will be given in Section 4.1.

The set of instantiations of the selected subset comprises a model of the situation. As discussed above, this selection process requires one to make two types of decisions, namely:

- What phenomena to model?
- How to model each of the chosen phenomena?

If the model includes the most detailed model fragment about every phenomenon that the library knows about, the resulting model will be the most comprehensive that could be produced from the library. However, in most situations, what is appropriate is much less than the most comprehensive model. For a model to be appropriate for a given problem, it must cover all the phenomena that are relevant for solving the problem. At the same time, it is desirable for the model to include only enough details about each phenomenon to produce a satisfactory answer. Therefore, the selection process must strive to include only the model fragments representing relevant phenomena described with just enough details. It must also make sure that the choice of model fragments be internally consistent in terms of the assumptions underlying them. For example, one cannot select a model fragment that assumes that an electrical signal propagates instantaneously to all parts of a circuit and another that assumes that it takes time. To summarize, the model formulation problem is that of deciding *what* to model and *how* to produce the *simplest adequate* model.

1.2. Our approach to model formulation

The primary type of task for which we target our model formulation work is simulation of time-dependent behavior. People perform such simulations for a variety of reasons, but in most cases the goal can be characterized as predicting how the values of a set of terms of interest change over time. Thus, in our work, we represent the user's query primarily as a set of terms of interest and assume that the goal of model formulation (and simulation) is to explain how their values change over time. Intuitively, predicting (and explaining) how values change over time requires the model to take into consideration all the things that could causally influence the term. Thus, the core high level mechanism driving our model formulation algorithm is backward chaining on all the possible causal influences on the goal terms. We start from each goal term, look for all the things that can causally influence the term, including objects and physical phenomena, decide what other terms can influence the goal term through those phenomena or objects, look for other things that can influence those terms, and so on recursively. For each phenomenon or object, we decide to include in the model, we determine how it should be modeled based on the set of modeling assumptions that are being maintained. The remaining sections of this paper give a detailed account of the representational and inferential mechanisms used by our method and analyze computational properties of the algorithm.

The contributions of our work can be summarized as follows:

- We enrich the representation of model fragment libraries by introducing additional constructs that enable statements about the relationships between model fragments in a library. The knowledge expressed by these statements make explicit assumptions that are implicit in the mind of the library builder, and thereby make the knowledge usable by a model formulation algorithm.
- We analyze the problem of model formulation as a problem of relevance reasoning and show how this analysis provides insight into the model formulation problem.
- Based on this analysis and the representational apparatus, we describe a novel model formulation algorithm for simulation.
- We define the concepts of simplicity and adequacy that make sense in the context of the model formulation problem. We present an analysis of our algorithm and show that it produces a simplest adequate model. We also show that the time complexity of the algorithm is polynomial with respect to the size of the problem and the model fragment library under a reasonable set of assumptions.
- Finally, we present experimental results demonstrating the effectiveness of our approach in limiting the size of formulated models.

Several pieces of work have addressed the model formulation problem for the compositional modeling approach [7,23,26]. Our work is distinctive in that it combines model formulation for simulation with guarantees of adequacy and simplicity.

1.3. Relevance reasoning for model formulation

Our model formulation procedure is based on a general framework for reasoning about relevance of knowledge. Formulating the problem of model construction as a type of relevance reasoning was instructive in teasing out the different parts of the model construction problem as well as providing guidance in treating each of the parts. Before explaining the connection between the problems of model construction and of relevance reasoning, we briefly explain the relevance reasoning problem.

Inference mechanisms in systems that have large and diverse knowledge bases encounter many irrelevant facts in the process of answering a given query. This causes them to explore many useless paths in their search, and therefore severely degrade their performance. Broadly speaking, there are two forms of irrelevance that may arise. The first is irrelevance of facts in the knowledge base. Certain facts may be shown not to contribute in any way to answering a query. However, if the inference mechanism cannot detect this irrelevance, it may still consider solutions that involve irrelevant facts. The second form of irrelevance is due to the level of detail at which the domain is conceptualized. For example, a knowledge base may identify properties of individuals at a granularity level that is too detailed for a given query, and as a result, the search space that needs to be explored is unnecessarily large.

Often, it is possible to detect efficiently that facts (or sets of facts) in the knowledge base are irrelevant to a query [16,21,30], or to detect that a knowledge base can be abstracted and still be able to answer a set of queries correctly [17,31]. In other cases, it is possible to give the system additional meta-level control advice as to which facts in the knowledge base are possibly relevant to a query, therefore enabling the inference mechanism to ignore the rest. Levy has developed a general framework for reasoning about relevance [16,18,21], and the model selection procedure presented in this paper is based on that framework. The framework provides a space of definitions of irrelevance with a comparison of their properties and provides a language in which additional meta-level knowledge about irrelevance can be given to a system. Levy also provides algorithms that for some forms of irrelevance efficiently decide which parts of the knowledge base are irrelevant to a query, thereby yielding significant speedups in performing inferences. An important aspect of relevance reasoning that is emphasized in this framework is the need to decide which facts in the knowledge base are relevant *without* actually considering the whole knowledge base. In particular, the algorithms described in [18] consider knowledge bases including a set of Horn rules and a set of ground facts, but the algorithms do not consider the ground facts when determining relevance of facts to a query.

In this paper, we argue that the model construction problem can advantageously be formulated as a problem of relevance reasoning. The first advantage of such a formulation is that it enables us to tease out the different parts of the model construction problem. In particular, the first aspect of the model construction problem is that we need to decide *which* phenomena can affect a given term. The second aspect is that we need to decide *how* to model each phenomenon. Note that these two decisions are not independent of each other. Finally, model construction must be done *without* knowing exactly which states the system may encounter.

Moreover, relevance reasoning also guided the treatment of each of these aspects of the problem:

(1) Determining which phenomena are relevant to the goal of explaining how a given term changes over time led to the high-level mechanism of our algorithm, which is backward chaining on all the possible causal influences on the term. In fact,

the graph of causal influences is similar in spirit to the *query tree*, the data structure developed by Levy to represent which facts may be part of a derivation of a query over a Horn rule knowledge base [18].

(2) Second, the need to reason about how to model each phenomenon led us to develop two representational tools for supporting such reasoning. The first tool (see Sections 3.1.1 and 3.2) provides the ability to state relevance claims that describe when a model fragment can be used and to express additional domain knowledge that comes to bear in selecting a model. Such claims are a generalization of the **consider** predicate described in [7], but Levy's relevance framework provided clear semantics for such statements. The second tool (see Section 3.1.4) enables the user to explicitly state the *difference* between the modeling assumptions made in alternative models of the same phenomenon. Such claims provide guidance in selecting the appropriate model of a phenomenon and in guaranteeing that the composed models are consistent with each other.

(3) Third, our algorithm must build a model without having complete information about the possible states of the system. Even given complete knowledge of the initial state, the future states are not known until we construct a model and perform the simulation. Thus, to build a model for simulation, we must somehow determine what could be relevant to answering a query without performing the simulation. Our algorithm chooses a model for simulation based only on knowledge of constraints on the possible states the system may enter, but without actually generating them. These constraints are encoded in the graph of causal influences that we use to guide the model construction algorithm.

1.4. Organization of the paper

The rest of this paper is organized as follows: Section 2 describes the basic knowledge representation and behavior prediction approach in compositional modeling, which we take as the starting point of our work on model formulation. Section 3 describes the additional knowledge and representation, the apparatus on which our formulation method relies. Section 4 defines the model formulation problem formally, describes our procedure, and presents an example of a model being formulated by the procedure. Section 4 also presents the results of our experiments with the procedures using several model fragment libraries. Section 5 analyzes the characteristics of the procedure and the models it generates. Section 6 discusses related work in model formulation as well as reasoning about relevance. Section 7 concludes with a discussion of some remaining research issues.

2. Compositional modeling

In this section, we describe our representation of physical knowledge and method for predicting behavior, which is based on the compositional modeling approach. Compositional modeling was first described by Forbus in his work on Qualitative Process Theory (QPT) [8] and is also the basis of subsequent work on qualitative modeling

```
Charge-sensitive-voltage-battery
    Participants: X: type Battery
    Quantities:
        voltage(X), charge-level(X), damaged(X)
    Conditions:
        ¬damaged(X)
        6 < charge-level(X) < 30
    Consequences:
        voltage(X) = f+(charge-level(X))
```

Fig. 2. An example model fragment.

by Falkenhainer and Forbus [7], Crawford, Farquhar and Kuipers [5], and Iwasaki and Low [11]. The purpose of this section is to explain in general terms the essence of the compositional modeling approach, which we take as the starting point in the work presented here on model formulation. We then describe the additional knowledge and the organization we impose on the knowledge base to facilitate model formulation. Though the terms used to describe parts of a model fragment and the actual simulation procedures differ somewhat in different systems [5,7,8,11], the underlying principles are the same in all of them. Here, we will use the nomenclature and the definitions given in CML (Compositional Modeling Language) [2]. CML was designed by the researchers of the compositional modeling systems mentioned above as a common model fragment representation language to enable sharing of knowledge bases. For a complete formal discussion of model fragments, see [3]. The description below includes only the aspects relevant to our discussion.

2.1. Model fragments

The basic idea in compositional modeling is to formulate a model of a given situation by putting together (composing) pieces of descriptions of physical phenomena in the domain. Each piece describes a conceptually independent aspect of some physical phenomenon, such as one aspect of a component behavior or a physical process. Each piece is a self-contained assertion whose applicability to a given situation is decided separately to generate the model of an entire situation. These pieces are called *model fragments*. For example, a model fragment may describe the dependence of the voltage of a battery on its charge level or may describe the process of fluid flow through a pipe connecting two containers. Fig. 2 shows an example of such a model fragment describing the relation between the charge-level of a battery and its voltage.

A model fragment names a set of *participants* in the phenomenon being described and a set of *conditions* which the participants need to satisfy in order for the phenomenon to take place. We say that an *instance* of a model fragment exists in a state when entities exist in the state for which the conditions for being participants in the model fragment are satisfied.[4] We also assume there is a set of unary predicates denoting *types* of entities in the domain. A model fragment also includes a set of *consequences* which specifies the behavior of the participant objects while the phenomenon is taking place.

We describe each part of the model fragment in detail below:

Participants. A name and type for each of the objects participating in a model fragment instance. A participant can be viewed as a unary function from a model fragment instance to an object in the domain. In Fig. 2, the participant is required to be an instance of class `battery`.

Quantities. Atomic expressions denoting time dependent attributes associated with the participants in a model fragment instance. Quantities can be continuous-valued functional expressions such as $voltage(X)$ and $current(X)$, discrete-valued functional expressions such as $color(X)$ and $location(X)$, or boolean-valued relational expressions such as $operational(X)$ or $damaged(X)$. The time argument of all quantities is left implicit.

Conditions. Statements that indicate when the phenomenon represented by the model fragment takes place by specifying constraints on the participants of the model fragment and on its quantities. The conditions include both structural constraints on the participants as well as constraints on the ranges of quantity values. We will sometimes use the term *operating conditions* to refer to these conditions to distinguish them from *modeling assumptions,* which are meta-level conditions that will be introduced in Section 3.1.1.[5] In our example, the model fragment's conditions require that the battery not be damaged and that the charge level of the battery be between 6 ampère-hours and 30 ampère-hours.

Consequences. Statements that are true whenever the phenomenon represented by the model fragment is taking place. Some of the consequences describe continuous phenomena (e.g., a fluid flow) by a set of equations involving the continuous quantities of the model fragment.[6] The equations may be quantitative (algebraic and ordinary differential equations) or qualitative (e.g., the rate of evaporation negatively affects the amount of water in the cup). Consequences can also be any other logical assertions that are true in a state in which an instance of the model fragment exists. Disjunctions are not allowed in the consequences. The consequences of the model fragment in our example asserts that the voltage and the charge level of a battery are qualitatively proportional (i.e. the partial derivative of one with respect to the other is positive).

One of the key assumptions regarding model fragments in the compositional modeling approach is that they be *composable.* In general, a quantity may be affected by more than one phenomenon at the same time. For example, the amount of water in a container

[4] In other words, a *model fragment instance* is a binding of each of the participants to an object in a particular state such that the objects satisfy the conditions in that state.

[5] The term "operating conditions" is also consistent with its usage in [7].

[6] To simplify discussion, we assume that the consequences do not contain inequalities on quantities. This does not impose a limitation on the representation since an inequality can be rewritten as a qualitative equation involving the difference.

can be affected by an evaporation process and by a condensation process. The way the amount actually changes over time is the combined effect of the two phenomena. In terms of representing such phenomena as model fragments, this means the following: As these two phenomena are independent of each other, they are represented by separate model fragments. Their consequences specify the effect each has on the quantity, the amount of water in the container, which is that they tend to increase (or decrease) the quantity. At simulation time, when all the model fragment instances that directly affect a quantity have been identified in a given state, their effects are combined into one complete equation under the closed-world assumption. The procedure for combining the effects is based on the semantics of the mathematical language for expressing consequences in model fragments.

The semantics of model fragments can be summarized as follows. Let $f_1, \ldots, f_n$ be the participants of a model fragment M. Let $o(X_1, \ldots, X_n)$ be the conditions of M and $b(X_1, \ldots, X_n)$ be the consequences of M. First, whenever a set of objects satisfies the conditions, there exists an instance of the model fragment. Formally:

$$\forall X_1, \ldots, X_n \left[o(X_1, \ldots, X_n) \Leftrightarrow (\exists\, m) M(m) \wedge \bigwedge_{j=1}^{n} f_j(m) = X_j \right]$$

The existence of the model fragment instance also implies that the quantities mentioned in it are defined. Furthermore, the existence of the instance implies that the consequences hold. Formally:

$$\forall X_1, \ldots, X_n \left[\left((\exists\, m) M(m) \wedge \left(\bigwedge_{j=1}^{n} f_j(m) = X_j \right) \right) \Rightarrow b(X_1, \ldots, X_n) \right].$$

2.2. Composing simulation models

Given a description of the physical configuration of a system and a particular state it is in, the model formulation task is to formulate a model that represents the physical phenomena occurring in the state. In compositional modeling, a model is composed of a set of model fragment instances whose conditions are satisfied in that state. Fig. 3 shows schematically the basic modeling framework in the compositional modeling approach. The set of all model fragment instances in a state comprise the *simulation model* for that state. The consequences of the model fragments in the simulation model give rise to a set of equations and logical formulas that hold as a consequence of the phenomena taking place. Those equations, if they include differential equations, determine how the state must be changing. The simulation model is given to the prediction engine, which generates the next state using the equations. If prediction is performed qualitatively, there may be a set of possible next states. Otherwise, there will be a unique next state. In either case, the conditions of the model fragments are re-examined in each of the next states to formulate a new simulation model.

In the basic modeling framework described thus far, the entire model fragment library is searched in every state to identify model fragments that can be instantiated. This presents two problems: First, the resulting model will be very complex, containing

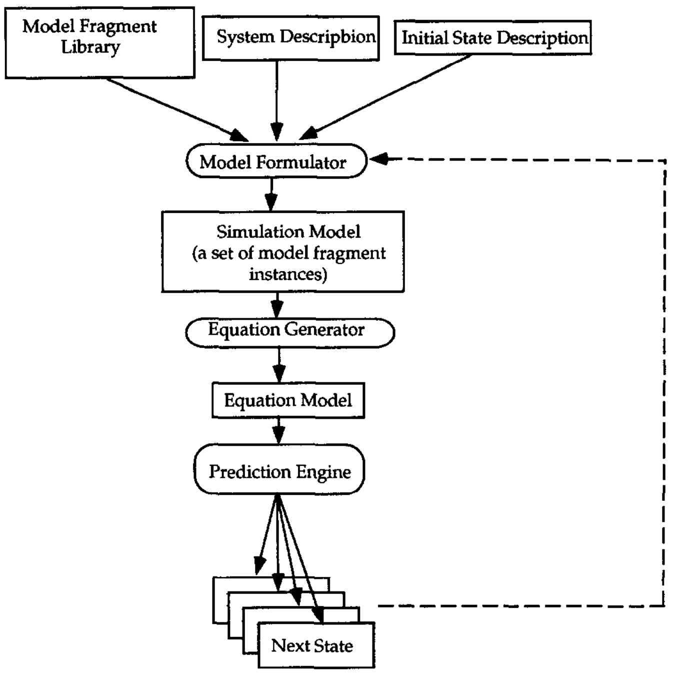


Fig. 3. Basic modeling framework in compositional modeling.

everything the system knows about the situation. This can not only make the simulation process expensive, but also the resulting prediction unnecessarily complex. Secondly, the resulting model may be inconsistent since there can be multiple model fragments in the library describing the same phenomenon based on different incompatible modeling assumptions. Therefore, it is necessary to limit the set of model fragments that will be considered for instantiation to a subset of the model fragment library that is sufficient for answering the query and makes a logically consistent set of modeling assumptions.

Selecting such a subset is difficult since, without performing simulation, one does not know exactly what will happen in the future states. The situation presents a quandary: To simulate, one needs to formulate a model. But, to formulate an appropriate model, one needs to know what can happen in the future, which requires simulation.

One extreme strategy for dealing with this problem is to perform relevance reasoning in every state. That strategy only requires determining what is relevant to a query in the current state, thereby eliminating the need to worry about what may happen in the future. However, that strategy can become inefficient as the model fragment library grows, since one must take the entire library into consideration in every state. The other extreme is to perform relevance reasoning only once at the beginning. That strategy will require model fragment selection only once, but since all legal states that the system *may* enter during the simulation must be considered, model selection will be more difficult and may produce a model that is unnecessarily complex. Any number of strategies between these two extremes can be devised, each with its run-time versus pre-processing time versus effectiveness tradeoffs.

The model formulation procedure that we will present in the following section enables one to pick a strategy between the two extremes that is appropriate for a given situation. The procedure provides a mechanism to infer what might happen in the future that can be relevant to the given goal in hand. It also provides a means for the user to make varying assumptions about what will not change in the situation to broadly delimit the scope of the phenomena that must be considered relevant.

Evaluation criteria. Before we proceed to describe our formulation method, we must discuss the evaluation criteria for such methods. A model formulation procedure is worthless unless the model produced is adequate for solving the given problem. What does adequacy mean in the context of simulation? For a model to be adequate, it must be internally consistent and also sufficient for answering the given query. In addition, we would like the model to be as simple as possible. Our model formulation procedure was devised with these goals in mind. We will provide a formal definition of the model formulation problem in Section 4.1, where we will also formally define adequacy and simplicity. In Section 5, we will use those definitions to prove that the procedure will in fact produce a consistent and sufficient model that is also a simplest model. We say *a simplest* instead of *the simplest* because there can be more than one simplest models for a given scenario according to our definition of simplicity. Furthermore, we will prove that under reasonable assumptions about the structure of the model fragment library, the time complexity of the procedure is polynomial. For now, we turn to the description of the organization of our model fragment library that facilitates model formulation.

3. Knowledge representation for model fragment selection

The previous section described model fragments and the idea of composing them to generate a model for a given situation. It also discussed the difficulty of actually formulating an appropriate model, especially for the purpose of simulation. In this section, we describe the additional representational tools we use in order to facilitate model formulation, such as composite model fragments, assumption classes (originally introduced in [7]), and explicit modeling constraints. The first two are used for imposing additional organization on the model fragment library, and the third is used for expressing domain-dependent knowledge about relevance of model fragments. Though these tools introduce more structure in the library, we argue that the designers of model fragments must have this structure in mind during the process of constructing a model fragment library. Here, we enable such designers to make explicit the assumptions underlying this structure, and thereby enable the model construction algorithm to benefit from the structure.

3.1. Model fragment library

In the basic compositional modeling framework described in Section 2.2, the model fragment library is simply a set of model fragments without any additional structure.

```
Charge-sensitive-voltage-battery
    Modeling Conditions:
        Rel(rechargeableBattery(X)) ∧
        Rel(charge-level(X)) ∧
        Rel(voltage(X))
```

Fig. 4. Modeling conditions for a model fragment.

However, if the content of the model fragments is examined, some model fragments are seen to be closely related. For example, some sets of model fragments simply provide different descriptions of the same phenomena, and some sets of model fragments describe the same phenomena in different operating regions. In order to facilitate model formulation, we introduce more organization into our model library that reflects such relations among model fragments. Model fragments are grouped into *composite model fragments* (CMFs), and CMFs are further grouped into *assumption classes*. Before we describe these concepts, we must first introduce modeling conditions and the notion of causal orientation of model fragments. Both are important in classifying model fragments into groups according to their contents. Modeling conditions make explicit the assumptions that underlie a particular description of a phenomenon as a model fragment. Causal orientation in a model fragment makes explicit the knowledge of possible causal relations implied by a model fragment.

3.1.1. Modeling conditions

In addition to operating conditions, we attach to model fragments another type of conditions called *modeling conditions*. Modeling conditions are used in order to distinguish alternative ways of modeling the same phenomenon. Modeling conditions are different from operating conditions in that they are conditions about the representation (i.e., meta-level conditions) as opposed to conditions on the domain and state.

We identify two classes of modeling conditions. The first class consists of relevance claims. As explained in Section 1.3, relevance claims can be used to express the assumptions underlying an abstraction. For example, a description of a battery that ignores its thermal aspects may be based on the relevance claim that the predicate *temperature* is irrelevant to the query. In modeling conditions, we state such relevance claims using the predicate `Rel`.

The second class of modeling constraints consists of assumptions about the problem solving task. These include assumptions about the desired accuracy of the answer and the temporal granularity of the model. For example, a model fragment describing the behavior of a battery over a few seconds would look quite different from one describing it over several days. The former would treat the voltage and the charge level as essentially constant quantities independent of each other. The latter would need to include the functional relation that exists between them.

Fig. 4 shows the modeling conditions for the *Charge-sensitive-voltage-battery* model fragment shown in Fig. 2. In the example, the model fragment requires that the charge level, voltage, and rechargeability of the battery be considered as relevant properties.

Formally, `Rel` is defined as follows. A simulator is given a model of an initial state and is said to produce a *simulation model* for each simulated state. The argument of predicate `Rel` can be either a ground term (i.e., a constant or functional term) or a ground atomic sentence.[7] $\mathtt{Rel}(\tau)$ is said to hold for a model M when τ occurs in M or in some simulation model M_s produced from M.

Note that as a consequence of the meaning of the predicate `Rel`, a modeling assumption that is a positive occurrence of `Rel` is a statement about what is included in a model, whereas an assumption that is a negative occurrence of `Rel` is a statement about what is excluded from a model (i.e., is a *simplifying* assumption).[8] For example, assuming $\neg\mathtt{Rel}(\mathit{Temperature}(\mathit{battery}))$ simplifies the representation because the temperature attribute of the battery can be ignored, whereas assuming $\mathtt{Rel}(\mathit{Temperature}(\mathit{battery}))$ requires that the temperature attribute of the battery be considered. We keep the same convention for other meta-level predicates that are used to specify properties of the resulting model. For example, the literal $\neg\mathit{large}(\mathit{timeScale})$ states that the representation is simplified to ignore longer term effects on the battery.

The predicate `Rel` is similar in spirit to the `Consider` predicate used in [7]. The main difference is that we provide clear semantics for `Rel` based on the relevance reasoning framework, and that we distinguish relevance of different elements (i.e., terms or ground atoms).

In general, it is difficult to assign semantics for relevance predicates (see e.g., [4, 9, 15]). However, in this paper we have limited the relevance claims to apply only to ground terms and ground atomic sentences. Hence, we avoid controversial situations (such as whether $\mathit{Rel}(p \vee \neg p)$ implies $\mathit{Rel}(p)$).

3.1.2. *Causal orientation of model fragments*

When building a model of a system and analyzing it, we often want to know exactly how the values of quantities are determined, i.e., what the causal dependencies are among the quantities in the model. For example, in a model containing Ohm's law, we may say that the voltage is determined by the current and the resistance. The *causal orientation* of the equations in a model determines the set of quantities that can be causally determined by that model.

Equations in model fragments describe the functional relations among the continuous quantities involved in the modeled phenomenon without specifying a particular causal direction. For example, the equation for Ohm's law, $V = iR$, only states the relationship between the current, the voltage and the resistance, without specifying which values cause which other values. The direction of causality only emerges when the equation is embedded in a system of equations each of which represents an independent mechanism and the quantities that are externally determined are specified. The theory of causal ordering [12] provides an operational definition of causality in such a system of equations.

[7] In order not to deal with second order sentences, we use KIF's quoting mechanism [10] when the argument is a ground atomic sentence.

[8] Though any finite model fragment implicitly makes a potentially infinite number of irrelevance assumptions (regarding all the terms not appearing in the model fragment), the only irrelevance assumptions that are useful to make explicit are those about the terms that do not appear in itself but do appear in an alternative, more detailed model fragment describing the same phenomenon.

By applying the causal ordering procedure to a self-contained system of equations, one can determine the causal dependency relations among the quantities in the equations. The result of the causal ordering essentially establishes a one-to-one mapping, $\Phi(e) = q$, from each equation, e, in the system to a quantity, q, that appears in the equation. The equation represents the mechanism that determines the value of the quantity, and all other quantities appearing in the same equation are the causal predecessors of the quantity. Since Φ is one-to-one, if $e_1 \neq e_2$, then $\Phi(e_1) \neq \Phi(e_2)$.[9] The quantities in a model that are not associated with any equation are called *exogenous*. They are the ones determined by some unspecified mechanisms external to the system being modeled.

Given a causal ordering Φ, we say that a quantity q_1 causally affects a quantity q_2 if:

- The quantity q_1 appears in the equation e, and $\Phi(e) = q_2$, in which case we say q_1 *directly* causally affects q_2, or
- There exists some quantity q_3, such that q_1 causally affects q_3 and q_3 causally affects q_2.

In compositional modeling, the causal orientation of equations can only be determined after the model fragments are instantiated and equations are assembled into a simulation model. Since an equation by itself is acausal, according to the theory of causal ordering, one cannot a priori specify for each model fragment the quantities whose values are caused by the model fragment. What one can say a priori about each model fragment is the possible set of quantities that could be determined by the model fragment. In general, this set can contain all the quantities mentioned in the consequences of the model fragment, but it could be a subset if some knowledge about the particular phenomenon represented by the model fragment allows one to limit the possible causal interpretations of the equations. We will call the set of quantities that can be determined by a model fragment its *output quantities*. The quantity that actually turns out to be causally determined by the model fragment (actually an equation in the model fragment) in any simulation model is always a member of that set.[10]

Furthermore, one can determine a priori for each member of the set of output quantities of a model fragment, the set of other quantities on which the quantity causally depends upon through the model fragment. These quantities must be determined by other model fragments in the model or must be exogenous to the model. Given a model fragment m and one of its output quantities, q, we can determine the set of all quantities that directly causally affects q through m as follows:

- The set includes all the quantities mentioned in the operating conditions of m.
- If q is a continuous quantity, the set also includes all the quantities appearing in the same equations as q in the consequences of m.

We call this set of quantities *the input quantities of m with respect to q*. Even though we have couched the discussion thus far in this section in terms of quantities and equations as if the consequences of model fragments consisted only of numerical

[9] We consider two equations to be equivalent if they contain the same quantities and are true for the same vectors of values of those quantities.

[10] We allow specification of output variables in a model fragment definition. However, such additional information is not necessary for our formulation algorithm to work correctly. Also, no such information was used in the empirical evaluation of the algorithm presented in Section 4.5.

quantities and equations, the concepts apply equally well to both numerical and non-numerical quantities and equations.[11] Thus, the input and output quantities can include any time-dependent attributes.

Given a set M of model fragments, one can draw a directed graph of causal influences as follows, where a node represents a quantity and an arrow represents a potential causal influence of the quantity at the tail on the quantity at the head:

(1) Draw an arrow from each q_i to q such that q_i is an input quantity of any member of M with respect to q.

(2) Repeat the procedure recursively for each q_i until no new arrows can be drawn.

Each path in this graph leading to q represents a path of potential causal influence of a quantity in M on q.

3.1.3. *Composite model fragments*

Some model fragments describe the same phenomenon, but differ only in their operating regions, i.e., the value ranges assumed for the continuous quantities in the model fragment. This happens often when a group of model fragments describe the behavior of the same device in different operating regions. For example, the functional relation between the voltage of a battery and its charge level changes depending on the value of the charge level: When the charge level is in a given range, say between 6 ampère-hours and 30 ampère-hours, the voltage is constant. On both sides of the range, the voltage tends to increase (or decrease) as the charge level increases (or decreases).

We group such model fragments into a single *composite model fragment* (CMF). A CMF is a set of model fragments describing the entire operating range of the quantities participating in the phenomenon. Consequently, all the members of a CMF must have the same set of participants and the same set of modeling conditions. The operating conditions of the model fragments in each CMF must guarantee that at most one model fragment from the CMF is included in any simulation model. Clearly, a CMF can be a singleton set. As we will show later, our model formulation procedure selects CMFs instead of individual model fragments.[12]

3.1.4. *Assumption classes*

Composite model fragments are further grouped into *assumption classes.*[13] An assumption class is a set of CMFs that describe the same phenomenon based on different and contradictory modeling conditions. As stated in Section 3.1.1, modeling conditions

[11] Recall the definition of quantities in Section 2.1 includes all atomic expressions denoting time-dependent attributes, including continuous-valued and discrete-valued functional expressions as well as boolean-valued relational expressions. Likewise, the concept of causal ordering applies to logical expressions [29].

[12] CMFs are in a sense an artifact of a particular restriction of the model fragment definition language we are using. If the model fragment definition language allowed disjunctions in the consequences, we would not need CMFs. In that case, our model formulation procedure would work with model fragments rather than with CMFs and achieve the same effect.

[13] We are giving the term "assumption-class" a slightly different meaning from that given to it by Falkenhainer and Forbus in [7]. In Falkenhainer and Forbus' work, an assumption class is a class of modeling assumptions which are assumptions about the same aspect of the represented phenomena but are mutually exclusive. For example, viscous fluid and inviscid fluid belong to the same assumption class, which is about the viscosity of fluid. In our usage, an assumption class is a set of mutually exclusive composite model fragments.

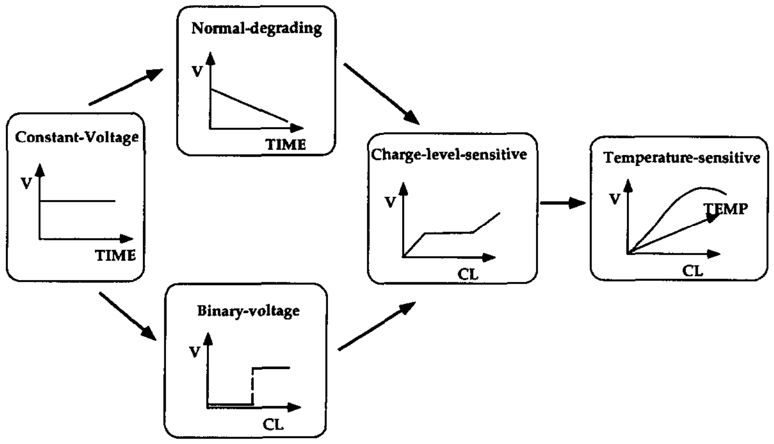


Fig. 5. Battery voltage assumption class.

express the assumptions made in the representation of the system. They express the underlying abstractions and approximations that are assumed by the model fragment. Fig. 5 shows an assumption class consisting of different ways of describing the voltage of a battery. One way to model the voltage is to assume it is always constant. Another way is to assume it steadily degrades over time. Yet another way of modeling a battery takes into consideration effects of the charge level and the temperature on the actual voltage produced. Since the modeling conditions of CMFs in an assumption class are mutually exclusive, any consistent set of modeling assumptions will include at most one CMF from a single instantiated assumption class.

Since CMFs in an assumption class all represent the same phenomenon, they each have the same set of participants. Within an assumption class, CMFs are partially ordered by a *simpler-than* relation which we denote by the predicate "$<$". A CMF c_i is said to be simpler than a CMF c_j (written $c_i < c_j$) if c_i makes a superset of the simplifying assumptions made by c_j. We will denote the transitive closure of $<$ by the predicate "$<^*$". In Fig. 5, the simpler-than relation is denoted by the directed arcs. There is an arc from c_i to c_j if $c_i < c_j$. We assume that every assumption class has a single most complicated CMF and a single simplest CMF. The most complicated CMF makes the fewest simplifying assumptions, and the simplest CMF makes the most simplifying assumptions. Finally, we assume that if $c_i < c_j$, then:

(1) The output quantities of c_j are a superset of the output quantities of c_i.
(2) If f_i is a causal ordering of the quantities of c_i, then there exists a causal ordering f_j of c_j such that the causal relations among quantities in c_i (given by f_i) are a subset of the causal relations among quantities in c_j (given by f_j).

All of these properties follow if we assume that whenever $c_i < c_j$, then c_i is an approximation of c_j of the type that Nayak calls *causal approximations* [24]. A model fragment is a causal approximation of another model fragment if the set of causal relations entailed by the former is a subset of those entailed by the latter. Nayak has argued that causal approximations cover most approximation relations encountered in practice.

We assume that the modeling conditions of CMFs in an assumption class *precisely* characterize the differences in assumptions made by CMFs in the assumption class. Specifically, this is formalized as follows: Let us denote as As_c the set of literals that are conjoined in the modeling conditions of the model fragments in a CMF c. Suppose $c_i < c_j$. Then we can annotate the link from c_i to c_j with a set of literals $p_1, \dots, p_n$, meaning that c_i is making the simplifying assumptions $\{\neg p_1, \dots, \neg p_n\}$ in addition to the simplifying assumptions made by c_j, or formally,

$$As_{c_i} = (As_{c_j} - \{p_1, \dots, p_n\}) \cup \{\neg p_1, \dots, \neg p_n\}.$$

The articulation of these differences will play an important role in our algorithm for selecting a simplest model. Finally, we say that an assumption class is *linear* if the $<$ is a total-order on the CMFs in the assumption class.

3.1.5. Coherence and completeness assumptions about the library

We make assumptions about coherence and completeness of the model fragment library in order for formulation of consistent and sufficient models to be possible. We discuss these assumptions in this section. After we describe our model formulation algorithm based on these assumptions, we will discuss the possibilities for relaxing them and the consequences of doing so.

Coherence of the library. The *library coherence* assumption essentially requires that if we have a set of model fragment instances whose modeling assumptions are consistent and whose operating conditions are satisfied in some state, then the resulting set of equations that model that state will not be over-constrained (i.e., will not have more equations than quantities). Formally, this assumption is defined as follows:

Definition 1. A model fragment library is said to be *coherent* if the following condition holds. Let M be a set of instances of model fragments from the library and s be any state such that:

(1) The modeling conditions, $a(M)$, of the model fragment instances in M are consistent,

(2) If $a(M) \models \mathtt{Rel}(\tau)$, then τ appears in the operating conditions or consequences of some model fragment instance in M, and

(3) The conjunction of the operating conditions of the model fragment instances in M are satisfied in s.

Then, the set of equations given by the union of the consequences of M are not over-constrained.

Note that a set of equations that are not over-constrained can always be made complete by assuming that some quantities are exogenous.

Completeness of the library. No library can be "complete" in the sense of covering all the knowledge of the world. There are things that any library, no matter how large and detailed, cannot explain. However, it is possible to define completeness with respect to known limitations of the knowledge contained in the library. The *library completeness* assumption states that the library contains knowledge about all the phenomena that can causally affect any quantity appearing in any model fragment in the library unless the quantity is explicitly known to be at the boundary of the library's knowledge. We will call the set $E_{global}(L)$ of quantities that are known to be at the boundary of knowledge contained in the library L *the globally exogenous quantities*. $E_{global}(L)$ is the set of all quantities such that they appear in model fragments but for which the library does not contain model fragments of all the phenomena that can directly causally affect the quantity.

Definition 2. A model fragment library is said to be *complete* if the following condition holds. Let A be a set of consistent modeling conditions, M be a set of model fragment instances, and s be any state such that:

(1) The conjunction of the modeling conditions, $a(M)$, of the model fragments in M is consistent,
(2) For any assumption class As that contains at least one CMF whose modeling assumptions are consistent with A, M contains an instance of a CMF from As,
(3) $a(M)$ is consistent with A,
(4) $a(M) \models \mathtt{Rel}(\tau)$ implies that τ appears in the operating conditions or consequences of at least one model fragment instance in M, and
(5) The conjunction of the operating conditions of model fragment instances in M is true in s.

Then,

- if the quantities in $E_{global}(L)$ are assumed to be exogenous, the set Eq_M of equations given by the union of the consequences of M is not under-constrained,[14] and
- for all v such that v appears in Eq_M and $v \notin E_{global}(L)$, M contains a CMF from each of the assumption classes representing phenomena that influence v under the conditions that are true in s.

Under this definition of completeness of a library, for any consistent state s, a complete library L will give rise to a set of equations that are sufficient to determine all the values of the variables appearing in them except for those in $E_{global}(L)$. We should note that this notion of completeness is orthogonal to that of "correctness" of a library. For example, a Newtonian physics library can be complete but will not give accurate results in states where objects are moving close to the speed of light.

3.2. Other modeling constraints

In addition to the modeling conditions attached to each model fragment, we assume a background theory of modeling constraints, $\mathcal{C}$. We use $\mathcal{C}$ to express additional constraints on the possible models. The constraints can either be domain independent (e.g., general constraints entailed by relevance claims) or domain specific constraints. The following

[14] Note that under the library coherence assumption, the set will also be self-contained.

constraint states that if the refinement of objects along the property r is relevant for an object O that is relevant to a query, and $r(O,X)$ holds, then X is relevant to the query:[15] For example, if Battery1 is relevant, the refinement of Battery1 along the property SubPart is relevant, and Chassis1 is a subpart of Battery1, then Chassis1 is a relevant object as well.

$$\mathit{RelObjectRefinement}(O,r) \wedge r(O,X) \wedge \mathit{RelObject}(O) \Rightarrow \mathit{RelObject}(X).$$

The following examples of constraints specify a variation on a heuristic used by Falkenhainer and Forbus [7]. Their heuristic is to include in a model all the components of the *minimal covering system*, defined to be the lowest common ancestor of the components mentioned in the query in the part-of hierarchy of the system being modeled. Falkenhainer and Forbus use this heuristic to assure that certain other objects be included in the model, given the initial set of objects of interest.

$$\mathit{structuralHierarchySlot}(p) \wedge \mathit{RelObject}(X) \wedge$$
$$\mathit{RelObjectRefinement}(X,p) \wedge p(X,Y) \Rightarrow \mathit{RelObject}(Y)$$

$$\mathit{structuralHierarchySlot}(p) \wedge p(X,Y) \wedge \mathit{RelObject}(X) \wedge \mathit{RelObject}(Y) \Rightarrow$$
$$\mathit{RelObjectRefinement}(X,p)$$

The above relevance axioms state that if the objects s_1 and s_2 are both relevant to the query, and t is their least common ancestor in the structural hierarchy, then any object in the hierarchy that is either between t and s_1 (or between t and s_2), or a child of such an object, will be considered relevant to the query.

Essentially, constraints can be expressed using arbitrary first order formulas. For efficiency reasons, we assume that the constraints in C are expressed using only Horn rules. Even though Forbus and Falkenhainer allow arbitrary clauses to state modeling constraints, Horn rules have been expressive enough for the modeling constraints we have so far encountered.

4. The model formulation method

In this section, we explain how the model formulation problem can be formulated as a problem of relevance reasoning, and we present the actual algorithm for formulating a scenario model. We also present a detailed example of models being formulated by the algorithm. As a basis for our discussion, we first formally define the problem of model formulation for simulation, given all the representational apparatus introduced in previous sections.

[15] Note that for clarity, we use predicate names for relevance that are specialized to the kind of entity being deemed relevant.

4.1. The problem definition

Informally, the model formulation problem is to choose a set of instantiated model fragments that can answer a query about a system given an initial state. However, a simulation of a system may go through many states, and, as we argued in Section 2.2, we do not want to repeat the costly selection process at every state. Therefore, we pose the model formulation problem as selecting a small set of *potential instances* of CMFs, called the *scenario model*. The scenario model has the property that its modeling conditions are consistent, and at every state, a simulation model can be easily composed from it.

Formally, we first define a *potential instance* of a CMF and then use that definition to define a scenario model.

Definition 3. A *potential instance* of a CMF is a pair $(c, plist)$, where c is a CMF and *plist* is a list each element of which is a potential binding of the form (p, o), where p is a participant in c[16] and o is an object that satisfies the type specification of p.

Note that the objects in the bindings of a potential instance of a CMF may or may not satisfy the statements in the Conditions part of the model fragments in the CMF. (Those conditions are checked when the simulation model is created for a specific state.)

Definition 4. A *scenario model* is a set of potential instances of CMFs whose modeling conditions are consistent.

Formally, the model formulation problem is to choose a scenario model, given a domain theory (i.e., model fragment library and background modeling constraints), a system description, and a query, defined as follows:

- *Scenario description*: A set of statements about a physical system and its initial state. These statements typically describe a set of individuals (i.e., components of the system), their physical structure, and the initial values of quantities related to those individuals.
- *Query*:
  - A quantity q (or list of quantities) whose value we want to predict by simulating the system.
  - A list E_{input} of exogenous quantities. The elements in E_{input} are assumed to be given and to be outside the scope of the simulation for which we are constructing a scenario model.
  - A list C_{input} of atomic ground sentences that must hold in all simulated states. These conditions are used to circumscribe the set of states for which we are creating a scenario model (e.g., if $\neg$`damaged(Battery)` is in C_{input}, then we construct a scenario model only for states in which the battery is not damaged).
  - A list *Init* of modeling constraints that we want to enforce. Implicitly, *Init* includes `Rel`(q).

[16] Recall that all model fragments in a CMF must have the same participants.

At each state during a simulation, the system checks the operating conditions *only* of the CMF instances in the scenario model. The operating conditions of at most one model fragment instance from each CMF will be satisfied in the state, and those model fragment instances will comprise the simulation model of the state. We denote the scenario model by $\mathcal{S}$ and the simulation model created from it in state s by $\mathcal{S}_s$.

In Section 2.2, we argued that the resulting scenario model must be adequate yet simple. Now, we can formally define the meaning of adequacy and simplicity in the context of the model formulation problem we have just defined. In our discussion, we use the phrase *logical-model* to refer to the standard notion of a model in logic (i.e., an interpretation that satisfies a set of formulas) and to distinguish it from a model of a physical device.

The first condition on a generated scenario model is that it must be adequate for answering the query. We define adequacy first for a simulation model and then for a scenario model as follows.

Definition 5. Given a set of exogenous quantities E, a simulation model M in a state s is *adequate* for determining a quantity q when the following conditions are satisfied:

C1 M contains a CMF instance from every assumption class a such that a can influence q in s and that there is at least one path of potential causal influence[17] from a variable in a to q that does not include a variable in E.

C2 The set of equations, Eq_M, in M determines the value of q uniquely; i.e.,

C2.1 Eq_M contains q, and

C2.2 Either:

C2.2.1 Eq_M contains a *self-contained* (i.e., not over-constrained or under constrained) subset that includes q, or

C2.2.2 Eq_M can be made self-contained by adding exogenous quantities that are in E.

We now define a scenario model as being adequate when it is coherent and sufficient as follows:

Definition 6. Given a set of background constraints $\mathcal{C}$, a scenario model $\mathcal{S}$ is adequate for answering a query $(q, E_{input}, C_{input}, Init)$ if:

D1 There is a logical-model L for $\mathcal{C} \wedge Init$ such that:

D1.1 The modeling conditions of all the CMFs in $\mathcal{S}$ are satisfied in L; and

D1.2 If $\texttt{Rel}(q_1)$ is satisfied in L for some quantity q_1, then q_1 occurs in some CMF in $\mathcal{S}$.

D2 For any state s of the simulation that does not contradict C_{input}, the simulation model $\mathcal{S}_s$ for s that results from $\mathcal{S}$ is adequate to determine q, assuming the quantities from E_{input} and E_{global} are exogenous.

Condition D1 ensures that the scenario model is coherent. D1.1 considers the modeling conditions, and D1.2 considers the relevance conditions. Condition D2 ensures that the

[17] See Section 3.1.2.

scenario model is adequate, i.e., that in every state, the resulting model is adequate for determining q.

In order for a scenario model to be useful, it should be as simple as possible:

Definition 7. A scenario model $\mathcal{S}_1$ is *simpler* than $\mathcal{S}_2$ if there is a mapping ϕ from the CMFs of $\mathcal{S}_1$ to the CMFs of $\mathcal{S}_2$ such that

(1) For every CMF $c \in \mathcal{S}_1$, $\phi(c)$ is from the same instantiated assumption class as c.

(2) Either $c = \phi(c)$ or $c <^* \phi(c)$.[18]

The mapping ϕ in the definition guarantees that $\mathcal{S}_1$ has no more CMFs than $\mathcal{S}_2$, and that for each CMF c in $\mathcal{S}_1$, there is CMF in $\mathcal{S}_2$ that is the same or more complicated than c.

The model selection problem is to find an adequate scenario model such that there is no simpler, adequate scenario model.

4.2. *Model formulation as relevance reasoning*

The approach to the model formulation problem advocated in this document is based on the intuition that several aspects of the problem are better viewed as problems of relevance reasoning. We explain our view and its import in this section.

Intuitively, we argue that the model formulation problem can be viewed as a combination of two subproblems. The first is to determine which phenomena (and therefore which quantities) are relevant to the query quantity. The second problem is to determine the level of detail at which to model the relevant phenomena. These problems are not independent of each other because the decision to model a certain phenomenon at a greater level of detail may require modeling additional phenomena. However, the kind of reasoning done for each one of these problems is different.

4.2.1. *Selecting the relevant phenomena*

The first part of the model formulation problem is to decide *which* quantities are relevant to the query quantity (and therefore, decide which phenomena should be modeled). Intuitively, a quantity u is relevant to the query quantity q if u can *causally influence* q, i.e., either (1) there is some possible state of the system in which u causally affects q, or (2) u can cause a change in the state of the system resulting in a state where some other quantity causally affects q (and, therefore, u indirectly affects the value of q). Consequently, finding the relevant quantities can be done by following the possible causal influences between quantities. The algorithm that we describe here traces through all the possible causal influences on the query quantity. The intuition underlying this algorithm is similar to the intuition underlying the construction of the query tree described in [21], which represents all the possible derivations of a query, and to other work on model formulation that follows causal chains in order to build models (e.g., [25,27,33]).

[18] Recall that the predicate $<^*$ is the transitive closure of the "simpler-than" predicate for CMFs.

4.2.2. Selecting the level of detail

The second part of the model selection problem is determining the level of detail at which to model each phenomenon. This entails deciding which abstractions and approximations can be made in modeling the system. Levy in [16,17] demonstrates that knowledge underlying such decisions can be stated as relevance claims and better understood when stated in that form. In our algorithm, we bring relevance knowledge to bear in choosing the level of detail in two ways:

- We articulate the *difference* between CMFs in an assumption class by the modeling constraints, expressed partially by relevance claims. Articulating the precise differences between the CMFs enables our algorithm to determine when to switch from one model fragment to another.
- Engineers have good general heuristics for selecting relevant detail when modeling physical systems. We use the modeling constraints C to express these heuristics declaratively (in the form of relevance claims) and to reason with them.

Our modeling algorithm uses both kinds of this knowledge to select a simplest scenario model.

4.2.3. Partial knowledge about the simulation states

Our algorithm selects a scenario model for a set of possible states of the system. Envisioning all the possible states that the system may reach beginning from the initial state is a very expensive operation [8] which we do not want to perform as part of the model formulation process. Therefore, our algorithm selects a scenario model based only on partial knowledge of the possible states. This knowledge is given implicitly by the set E_{input} of quantities that are assumed to be exogenous and by the time invariant facts in the description of the system. The problem we face here is analogous to the relevance reasoning problem discussed by Levy and Sagiv in [20] in the context of Horn rule knowledge bases and database systems. There, the problem was to decide which ground formulas are irrelevant to a given query without actually knowing which ground formulas are in the knowledge base. In doing so, unless it was explicitly stated that some ground formulas are *not* in the knowledge base, it was assumed that they *might* be.[19] Analogously, here we assume that the system can actually reach any state that is consistent with our partial knowledge. As in [20], any additional knowledge about the reachable states may enable us to select a simpler scenario model. Assuming partial knowledge about the world (and the knowledge base) is a key aspect in making relevance reasoning practical.

4.3. Model formulation algorithm

Based on the discussion in the previous sections, we now describe our model formulation algorithm. Informally, the algorithm follows all possible influences on the query in order to find all the quantities that can affect the query. For each such quantity, the

[19] Note that the fact that a certain formula might be in the knowledge base may affect the relevance of another formula.

algorithm selects the simplest CMF that describes it such that the set of selected CMFs make consistent modeling assumptions. The details of the algorithm are shown in Fig. 6.

To find all the quantities that can affect the query q, the algorithm begins by considering all the assumption classes of which the quantity q may be an output quantity. The set of output quantities of an assumption class is the union of all the output quantities of the model fragments contained in the assumption class.

From each such assumption class, we select one CMF and recursively consider all the input quantities of the CMF with respect to q. The recursion bottoms out when we reach the exogenous quantities given in E_{input} or E_{global}.

To select a CMF from an assumption class, we maintain a list, *Rel*, of modeling assumptions made thus far about the model. The list initially includes the assumptions given in *Init* (and in particular, the relevance of the query q). At every step, we choose the simplest CMF that does not *contradict* the assumptions in *Rel*.

Adding a new CMF to the scenario model may imply that we add assumptions to *Rel*, and that we need to revise previous choices of CMFs. We perform adjustment steps (via the while-loop in **select-scenario-model**) until all the choices of CMFs are consistent. In what follows, we illustrate the execution of the algorithm with an example.

4.4. Example

The example is a simple circuit containing a solar array (`SA1`) and a rechargeable battery (`BA1`), shown in Fig. 1. Fig. 7 shows the scenario description and Fig. 8 shows the model fragments in the domain theory library. For each CMF in the domain theory, the CMF's behavior equations (consequences) and the list of quantities appearing in its operating conditions are shown. The annotated assumption classes are shown in Fig. 9. The query is `Voltage(BA1)`, with a list of exogenous quantities which includes all the quantities mentioned in the scenario description except `Damaged(BA1)`. The set of modeling constraints is empty.

The query quantity `Voltage(BA1)` is the only item on the queue initially, and so the algorithm identifies `Battery-voltage-ac(BA1)` as an assumption class that can affect it.[20] To select a CMF out of this assumption class, we start from the simplest, `Constant-voltage-CMF`. Since there are no earlier modeling assumptions, this choice is consistent, and we select this CMF. This choice results in the addition of the following to our modeling assumption list, *Rel*:

`Rel(Battery(BA1))` and `Rel(Damaged(BA1))`.

Since the quantity `Voltage(BA1)` can be influenced by the quantity `Damaged(BA1)` (through the CMF `Constant-voltage-CMF(BA1)`) which is not exogenous, the quantity `Damaged(BA1)` is placed on the queue and becomes the new current goal. We find the assumption class `Battery-damage-due-to-overcharge-ac` that can affect `Damaged(BA1)`, out of which `Battery-damage-CMF` is selected since it is the only member. This selection causes the literals:

[20] In our implementation of the algorithm, we make use of a data structure that enables us to efficiently find the assumption classes that affect a given quantity without searching the whole model fragment library.

```
procedure select-scenario-model($q$, $E_{input}$, $C_{input}$, $Init$, $C$)
/* $q$: the query quantity. */
/* $E_{input}$: the list of exogenous quantities given in the query. */
/* $C_{input}$: the list of conditions that must hold in all simulation states given in the query. */
/* $Init$: the list of modeling constraints given in the query. */
/* $C$: the background theory of modeling constraints. */
/* $Q$: a queue of quantities and terms. */
/* $Rel$: the current list of modeling constraints. */
/* $Model$: the scenario model being constructed. It is a list of pairs $(c, x)$, */
/*   where $c$ is a potential instance of a CMF */
/*   and $x$ is a quantity or a term that $c$ could causally affect. */
/* $As_c$: the modeling assumptions of CMF $c$. */
begin
  $Q = \{q\}$.
  $Rel = Init$.
  $Model$ = nil.
  repeat
    $q_1 = dequeue(Q)$.
    $As$ = assumption classes in which $q_1$ can be an output quantity and
      whose operating conditions do not contradict $C_{input}$.
    for each $a \in As$ do:
      select-from-assumption-class$(a, q_1)$.
      while there is a pair $(c, q') \in Model$ and $p \in Rel$ such that $\neg p \in As_c$
        remove $(c, q')$ from $Model$.
        select-from-assumption-class$(A_c, q')$.
        /* $A_c$ is the assumption class from which $c$ was chosen */
  until $Q$ is empty.
  return the set $\{c \mid (c, q') \in Model\}$.
end select-scenario-model.

procedure select-from-assumption-class$(A, q)$
/* $A$ is a potential instance of an assumption class that can determine $q$. */
begin
  $c$ = A simplest CMF in $A$ such that there does not exist $p$ such that $\neg p \in As_c \land p \in Rel$.
  $Model = Model \cup \{(c, q)\}$.
  $inputs$ = the union of:
    The quantities that appear in equations with $q$ and
    The quantities in the operating conditions of $c$.
  for every $X \in inputs$ do
    if $X$ has not been in $Q$ and $X \notin E_{input}$ then
      enqueue $X$ onto $Q$.
  $Rel = DeductiveClosure(C \cup Rel \cup Pos(As_c))$.
  /* $DeductiveClosure$ returns the set of ground atomic formulas derivable from its argument. */
    $Pos(As_c)$ is the list of positive literals in $As_c$.
  if $\mathtt{Rel}(q_1) \in Rel$ and $q_1 \notin E_{input}$ and $q_1$ has not been in $Q$ then
      enqueue $q_1$ onto $Q$.
end select-from-assumption-class.
```

Fig. 6. Model selection algorithm.

Scenario description	Legend
Solar-array(SA1)	*CL*: Charge-level(X)
Battery(BA1)	*V*: Voltage-produced(X)
Rechargeable(BA1)	*TEMP*: Temperature-of(X)
Plus-terminal(BA1)=t4	*I*: Current(Plus-terminal(X))
Minus-terminal(BA1)=t3	*DOD*: Average-depth-of-discharge(X)
Plus-terminal(SA1)=t2	*TSLC*: Time-since-last-conditioning(X)
Minus-terminal(SA1)=t1	
Electrically-connected(t2,t4)	
Electrically-connected(t1,t3)	
¬Damaged(BA1)	

Fig. 7. The initial state of the system.

CMF	Behavior	Quantities in operating cond.
Battery-voltage assumption class:		
Constant-voltage-CMF	$V = C_0$	Battery(X), ¬Damaged(X)
Binary-voltage-CMF	$V = \begin{cases} 0 & \text{if } CL < c_0 \\ v_1 & \text{if } CL \geqslant c_0 \end{cases}$	Battery(X), ¬Damaged(X)
Normal-degrading-CMF	$V = f(\mathit{Time})$	Battery(X), ¬Damaged(X)
Charge-sensitive-CMF	$V = f(CL)$	Battery(X), ¬Damaged(X), Rechargeable(X)
Temperature-sensitive-CMF	$V = f(TEMP, CL)$	Battery(X), ¬Damaged(X), Rechargeable(X)
Battery-charge-level assumption class:		
Constant-charge-level-CMF	$CL = c_1$	Battery(X), ¬Damaged(X)
Normal-accumulation-CMF	$CL = \int I\,dt$	Battery(X), ¬Damaged(X), Rechargeable(X)
Accumulation-with-aging-CMF	$CL = \int I\,dt - f(DOD, TSLC)$	Battery(X), ¬Damaged(X), Rechargeable(X)
Battery-damaged-due-to-overcharge assumption class:		
Battery-damage-CMF	*Damaged*(*X*)	Battery(X), ¬Damaged(X), Rechargeable(X) CL(X)

Fig. 8. Model fragments for the battery example.

`Rel(Rechargeable(BA1))` and `Rel(Charge-level(BA1))`

to be added to *Rel*. The tree of quantities and assumption classes together with the current content of *Rel* at this point is shown in Fig. 10. The CMF in bold face is the one currently selected. The node marked X is in E_{input} and, therefore, is not expanded any further.

The addition of the above literals to *Rel* makes the assumption list inconsistent since both ¬`Rel(Rechargeable(BA1))` and ¬`Rel(Charge-level(BA1))` were assumed by `Constant-voltage-CMF`.

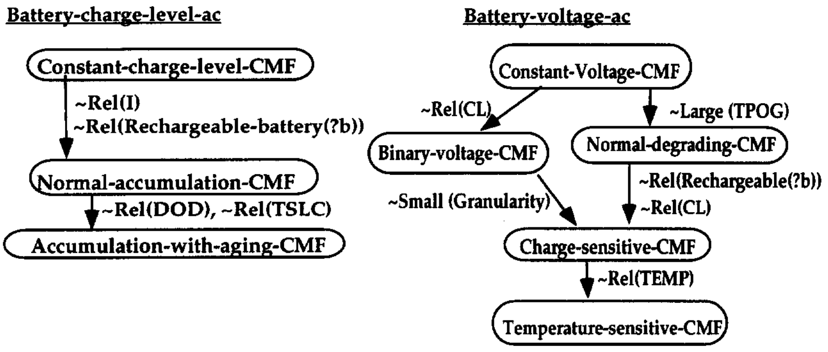
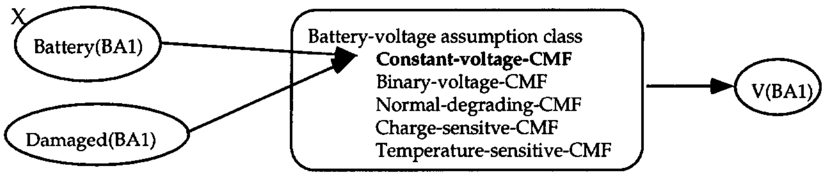
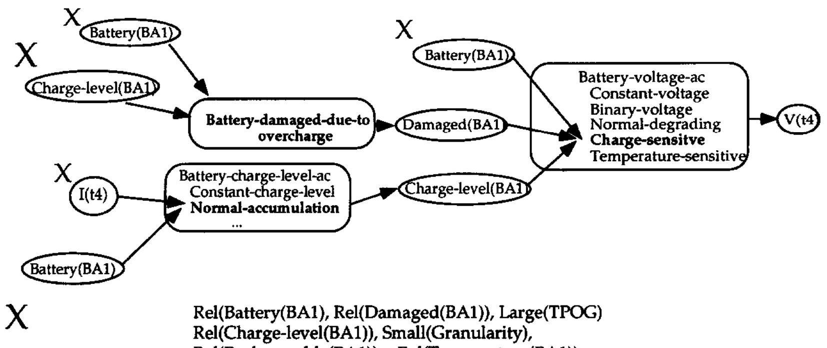


Fig. 9. Assumption classes.



Rel(Battery(BA1), Rel(Damaged(BA1)), ~Large(TPOG)
~Rel(Charge-level(BA1)), ~Small(Granularity),
~Rel(Rechargeable(BA1)), ~Rel(Temperature(BA1)

Fig. 10. An intermediate goal tree in Example 1.



Rel(Battery(BA1), Rel(Damaged(BA1)), Large(TPOG)
Rel(Charge-level(BA1)), Small(Granularity),
Rel(Rechargeable(BA1)), ~Rel(Temperature(BA1)),
~Rel(DOD), ~Rel(TSLC)

Fig. 11. The final goal tree of the example.

Table 1
Reduction in the size of models

Example name	Max. formulated model size	Max. possible model size
SC	7	31
EPS	28	41
RCS	252	418
BMS	18	212

To resolve the inconsistency, we adjust the choice of `Constant-voltage-CMF`, and we now select `Charge-sensitive-CMF`, which is the simplest CMF that does not contradict the current modeling assumptions.

The current goal quantity now becomes `Charge-level(BA1)`. The assumption class that can affect this quantity is `Battery-charge-level-ac`, and we select from it the CMF `Normal-accumulation-CMF(BA1)`, which is the simplest CMF that is consistent with the current modeling assumptions. `Current(Plus-terminal(BA1))` can influence `Charge-level(BA1)` through this CMF. However, since it is an exogenous quantity, it is not placed on the queue. The queue is now empty and the procedure terminates. The final tree of quantities and assumption classes is shown in Fig. 11. The resulting scenario model contains:

```
Charge-sensitive-CMF(BA1),
Battery-damage-due-to-overcharge-CMF(BA1),
Normal-accumulation-CMF(BA1).
```

and makes the following modeling assumptions:

```
Rel(Battery(BA1)),  Rel(Damaged(BA1)),
Rel(Rechargeable(BA1)),  Rel(Charge-level(BA1)),
Large(TPOG),Small(Granularity),
¬Rel(Temperature(BA1)),  ¬Rel(DOD),  ¬Rel(TSLC)
```

Note that the procedure terminates at this point in the example because the quantity `Current(Plus-terminal(BA1))` was specified as exogenous in the query. Had it not been specified exogenous, the procedure would have added more CMFs to the model, including those representing other components and processes affecting the current.

4.5. Experimental results

We have conducted experiments with the model formulation algorithm on several domain theories. The purposes of the experiments are to test our implementation of the algorithm and to verify its effectiveness with actual model fragment libraries.

The results of the experiments generally confirm the effectiveness of the formulation algorithm in limiting the model size. For each example, the model sizes varied widely depending on queries. Table 1 shows the size of the largest model formulated along with

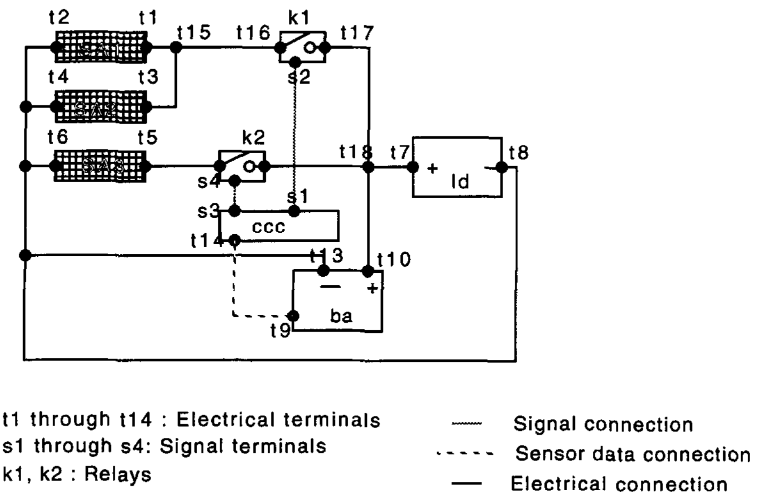


Fig. 12. Electrical Power System.

the size of the maximum model that would be created without the algorithm for each of the four domain theories tried. Since model sizes can be made arbitrarily small by adding a priori exogeneity assumptions to the query, the model sizes shown are those obtained without any such assumptions. Even though all the model fragment libraries used were fairly narrowly focused on certain aspects of one device, the results show that a significant reduction in the size of the model was achieved for most queries. The table shows that reduction in the size of formulated models ranged from 32% (EPS) to 91% (BMS). We expect that reduction in model sizes would be even more significant for more general-purpose model fragment libraries with a broader scope.

We briefly describe the four example domain theories used to test the algorithm:

Simple circuit (SC). The circuit shown in Fig. 1 used as the illustrative example in Section 4.4. The model fragment library contains 32 model fragment definitions.

Electrical power system (EPS). The Electrical Power Supply system of a satellite shown in Fig. 12. The system consists of a series of solar arrays, relays, an electrical load, a rechargeable battery, a charge current controller, and their connections. While the satellite is in the sun, the solar arrays supply power to the load while recharging the battery. The battery supplies power in eclipse. The charge current controller monitors the charge level of the battery and opens and closes the relays in order to prevent the battery from damage due to over-charging. The model fragment library contains 66 definitions.

Reaction control system (RCS). The Reaction Control System of the Space Shuttle shown in Fig. 13. The system consists of two structurally identical subsystems for supplying oxygen and fuel. Oxygen and fuel, separately propelled by pressurized helium, meet in the thrusters and ignite, generating thrust. The model fragment library containing 95 definitions is mainly concerned with the fluid dynamic behavior of the system.

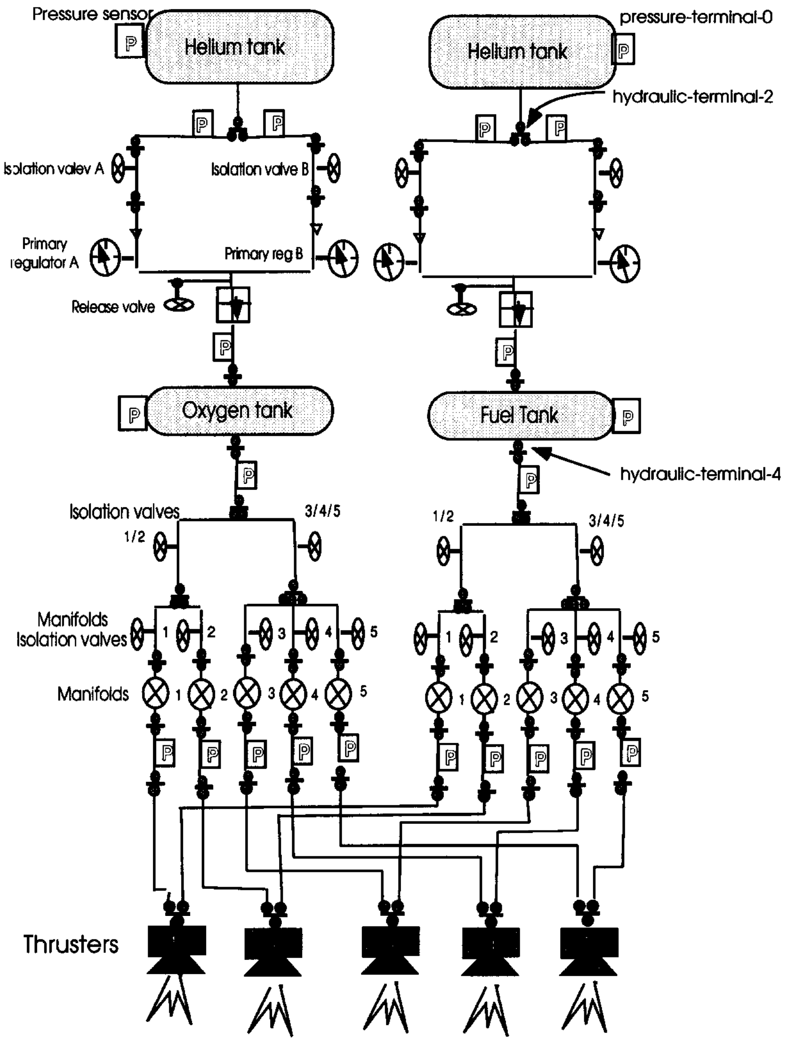


Fig. 13. Reaction Control System.

Bimetallic-strip temperature gauge (BMS). The temperature gauges shown in Fig. 14. It is described by Macaulay in [22]. The device is intended to measure the temperature of liquid. The resistance of the thermistor changes according to the temperature of the liquid, which results in change in the current through the wire. The current determines the heat generated by the coil, which changes the temperature of the bimetallic strip. As the bimetallic strip is heated by the coil, it bends, rotating the pointer. The model fragment library contains knowledge about kinematics, electricity, thermodynamics, thermal conduction, as well as the behaviors of the thermistor and the bimetallic strip. The library contains 80 definitions.

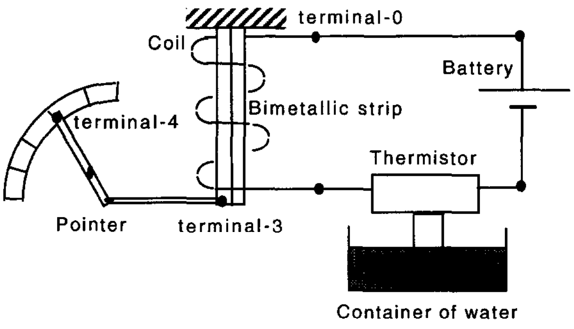


Fig. 14. Bimetallic strip temperature gauge.

Table 2
Size of the examples

Example	Library size	Scenario size	Max. scenario model size
SC	32	11	31
EPS	66	54	41
RCS	95	289	418
BMS	80	33	212

Except for the simple circuit example, which was constructed explicitly for the purpose of demonstrating the algorithm, the domain theories had been originally constructed independently to model the behavior of respective devices. The last example of the bimetallic strip temperature gauge knowledge base had been originally constructed by Nayak to test his approach to model formulation in his thesis, which extensively uses the bimetallic strip temperature gauge as an example [23]. We rewrote his model fragment library in CML for the experiment.

Table 2 shows the size of each example, in terms of the size of the model fragment library, the size of the scenario, and the maximum possible size for a scenario model. The size of the model fragment library is measured by the number of model fragment definitions in the library. The size of the scenario is the total number of components in the scenario.[21] The maximum possible size for a scenario model is the number of CMF instances that would be created if all the knowledge in the library were used to formulate a model. In other words, it is the upper limit on the size of a scenario model given the library and the scenario.

Tables 3–6 show quantitative data from each example. For each domain theory, the tables show the goal term, the formulated model size (in terms of the number of CMF instances in the model), the number of goals explored by the formulation procedure, and the numbers of facts and components determined relevant by the procedure. In Table 5,

[21] In these domain theories, a terminal or a junction, which represents a connection among terminals, is modeled as one entity.

Table 3
Sample results from the SC example

Goal term	Model size	Goals	Relevant facts	Components
stored-charge(BA1)	3	6	13	4
drained-p(BA1)	4	7	14	4
expected-charge-level(BA1)	8	18	29	8
electromotive-force(BA1)	7	16	26	8
damaged-p(BA1)	7	15	26	8
voltage-at-terminal(t4)	7	16	26	8
current-thru-terminal(t4)	7	16	26	8
voltage-at-terminal(t3)	1	4	7	3

Table 4
Sample results from the EPS example

Goal term	Model size	Goals	Relevant facts	Components
orbit-time(sun)	2	2	5	2
sun-shining-p(sun)	2	2	5	2
signal-on-at-terminal-p(s1)	28	98	143	43
relay-closed-p(k1)	28	98	143	43
voltage-at-terminal(t1)	28	98	143	43
current-thru-terminal(t1)	28	98	143	43
voltage-at-terminal(t7)	28	98	143	43
current-thru-terminal(t7)	28	98	143	43
voltage-at-terminal(t10)	28	98	143	43
current-thru-terminal(t10)	28	98	143	43
voltage-at-terminal(t13)	1	10	18	7
voltage-at-terminal(t14)	28	98	143	43

Table 5
Sample results from the RCS example

Goal	Model size	Goals	Relevant facts	Components
sensed-pressure-at-terminal(pressure-terminal-1)	20	84	52	20
*sensed-pressure-at-terminal(pressure-terminal-1)	20	84	52	20
pressure-at-terminal(hydraulic-terminal-2)	252	769	936	267
pressure-at-terminal(hydraulic-terminal-4)	252	758	936	267
thruster-cavitating-p(thruster-1)	252	760	936	267
*thruster-cavitating-p(thruster-1)	136	432	508	141
thruster-cavitating-p(thruster-5)	252	736	936	267
*thruster-cavitating-p(thruster-5)	136	420	508	141

Table 6
Sample results from the BMS example

Goal	Model Size	Goals	Relevant Facts	Components
resistance(Coil-0)	1	1	3	2
current-through-terminal(terminal-0)	15	24	38	5
temperature-of(Bimetallic-Strip-0)	15	26	38	5
temperature-of(Coil-0)	15	26	38	5
heat-flow(Coil-0)	15	26	38	5
input-electrical-power(Coil-0)	15	26	38	5
position(terminal-3)	18	39	59	10
resistance(Thermistor-0)	12	21	34	5
voltage-difference(Thermistor-0)	12	23	34	5
voltage-difference(Coil-0)	15	24	38	5
voltage-difference(Battery-0)	2	6	13	4
temperature-of(Thermistor-0)	12	23	34	5
heat-flow(Bimetallic-Strip-0)	15	26	38	5
self-induced-emf(Coil-0)	15	24	38	5
angular-position(terminal-4)	18	39	59	10

the goals with a * indicate the cases where the given query included a set of exogenous variable assumptions. Those assumptions were essentially intended to broadly limit the scope of the model to either the oxygen or the fuel side of the RCS. In all other cases, the query did not include any exogenous variable assumptions (i.e. $E_{input} = \emptyset$).

Our goal for constructing the BMS example was to obtain a direct comparison with the result obtained by Nayak for the same example. However, because of the different way in which goals are formulated in Nayak's system, our program initially did not formulate the same model as the one produced by Nayak. In his approach, the query to the model formulation system is formulated as a statement of the expected causal relation, which one wishes to establish by formulating a model. For example,

(causes (temperature thermistor-1) (angular-position pointer-2))

is an expected causal behavior given as a query to the system. In contrast, a query in our formulation of the problem specifies only the term (or terms) to be causally explained by a model. In other words, our formulation only allows the *effect* term the model is to explain to be specified in the query but not a *cause* term. Thus, in the case where *(angular-position pointer-2)* was given as the goal term in the query, our algorithm produced a much smaller and simpler model than that produced by Nayak's system. The model produced only included the pointer and the bimetallic strip, where the bimetallic strip was modeled as being in thermal equilibrium. This is a perfectly valid model in the absence of any a priori relevance assumptions. Once we provided assumptions about relevance of heat flow into the bimetallic strip, our algorithm produced a model that included all the components and their behaviors that are relevant and sufficient for explaining the possible causal relation between the two terms.

One notable advantage of our approach over Nayak's is the fact that our formulation does not rely on heuristic rules as Nayak's system does in order to formulate a model. Nayak's approach depends on manually formulated heuristic rules such as *Component Interaction Heuristic* and *Heuristic Coherence Constraints* for model formulation. As our approach does not require any such heuristics, we ignored all the heuristic rules when constructing the CML model fragment library of the BMS example from Nayak's original domain theories. However, if one wishes to provide domain- or problem-specific heuristics to guide the model formulation process, it is easy to do so in the form of additional relevance assumptions and modeling constraints.

5. Analysis

In Section 2.2, we discussed the evaluation criteria for model formulation techniques, and argued that a model formulation procedure must produce an adequate yet simple model. How does the algorithm we presented in this document fare against these criteria? In this section, we prove that our algorithm produces a simplest adequate scenario model for answering a query. We discuss the assumptions under which this result holds and the consequences of relaxing them. The following theorem establishes the main properties of the algorithm:

Theorem 8. *Let $\mathcal{L}$ be a library of model fragments describing a domain, and $\mathcal{C}$ be a set of modeling constraints. Let $(v, E_{input}, C_{input}, Init)$ be a query about a system. Let $\mathcal{S}$ be the scenario model resulting from algorithm* **select-scenario-model**. *Furthermore, assume that:*

T1 *$\mathcal{L}$ satisfies the library coherence and completeness assumptions*;

T2 *If c_i and c_j are two CMFs in an assumption class in $\mathcal{L}$ such that $c_i < c_j$, then c_i is a causal approximation of c_j*;

T3 *All modeling constraints in $\mathcal{C}$ are either ground atomic formulas or Horn rules*;

T4 *The most complicated scenario model, defined to be all the possible instantiations of CMFs that are the most complicated in their assumption class, is adequate for answering the query*;[22] *and*

T5 *All assumption classes in L are linear.*

Then, $\mathcal{S}$ is an adequate scenario model for $(v, E_{input}, C_{input}, Init)$, and there is no scenario model $\mathcal{S}_1$ such that $\mathcal{S}_1$ is simpler than $\mathcal{S}$.

The complete proof, given in the Appendix, shows first that the algorithm terminates. It then shows that the output of the algorithm satisfies the conditions of adequacy and simplicity given in Definitions 6 and 7.

We can also show that $\mathcal{S}$ is built in time that is polynomial in the size of the problem, as stated in the following theorem:

[22] Note that the most complicated scenario model needs to include only the *valid* instantiations of model fragments, i.e., instantiations in which the objects satisfy the type conditions in the model fragment and the time invariant facts in the description of the system.

Theorem 9. *Let d be the maximum number of CMFs in an assumption class, and let n be the number of instantiated assumption classes in S. Let l be the sum of the size of C and the number of ground atoms that appear in the instantiated modeling conditions of CMFs in S. The running time of finding S is polynomial in n, d and l.*

Proof. Since the modeling constraints are Horn, computing the logical closure of the set of modeling assumptions is done in time polynomial in l. This is done every time we call the procedure **select-from-assumption-class**. The number of times this procedure is called is at most nd. This can be seen by observing that every call to **select-from-assumption-class** may, at worst, replace a CMF by another one that is more complicated than it. Since there are n instantiated assumption classes in S and at most d CMFs per class, this can only be done nd times. Consequently, the overall running time of the algorithm is polynomial in n, d and l. □

5.1. Relaxing the assumptions

In this section, we discuss the effect of relaxing some of the assumptions made in Theorem 8.

5.1.1. The library coherence assumption

The most significant assumption that we made is the library coherence assumption. Although the assumption may seem too strong, there is a compelling argument for it. Specifically, if the assumption does not hold, there is a problem with the model fragment library. If there is a set of model fragments that satisfy the modeling constraints but give rise to an over-constrained set of equations, that is an undesirable feature of the library that calls for additional knowledge acquisition. It should be noted that the library coherence assumption is made implicitly in [7]. In fact, if we assume (as in Qualitative Process Theory [8]) that all equations are uniquely causally oriented, then the library coherence assumption follows when we make the causal approximations assumption and the assumption that the most complicated scenario model is adequate.

We can relax the library coherence assumption at the cost of doing more work at every state of the simulation. Specifically, in the absence of the coherence assumption, the scenario model created by our algorithm is guaranteed to produce a set of equations at every state from which a complete model can be extracted (perhaps by removing some equations). We can extract the complete model efficiently using the methods described by Nayak in [24].

5.1.2. Causal approximations and Horn restriction

The only role of the causal approximations assumption and the restriction that the modeling constraints must be Horn is to guarantee efficient performance of the model formulation algorithm. The causal approximations assumption guarantees that when we select a more complicated CMF in an assumption class that the number of elements in the set of simplifying modeling assumptions decreases (i.e., more positive literals are added to *Rel*). The Horn restriction guarantees that once a positive literal has been put in *Rel* that it will not be retracted. Relaxing either of these two assumptions will require the algorithm to perform arbitrary backtracking and constraint satisfaction. As shown in [24], this will cause the model selection problem to be intractable. Finally, the assumption that all assumption classes are linear guarantees that a simplest scenario model can be found in polynomial time. If the assumption does not hold, the algorithm, as described, will still work in polynomial time but may not produce a simplest model. It is possible to modify the algorithm such that it finds a simplest model, but then it will not run in polynomial time.

6. Related work

Several researchers have considered the problem of model formulation. Their work addresses one or both of the two aspects of the model formulation problem, namely model construction and model simplification.

Nayak [23] addressed both aspects. Nayak describes an algorithm for constructing a model for the single state case. His algorithm also follows possible causal influences; however, these influences must be given explicitly using the *component interaction heuristic*. In contrast, our work exploits the structure of the model fragments to derive these links, thereby not burdening the user with the error prone task of putting them in. It should be noted, however, that user intervention, as in Nayak's scheme, can enable a further focusing of the search by inserting only a subset of the links.

In choosing a model fragment from every assumption class, Nayak chooses the most complicated one, and then uses a procedure to simplify the resulting model. Our algorithm builds the model by selecting the simplest CMF possible in every class and only adjusts the choice if necessary. In cases where the CMFs in an assumption class vary significantly in their complexity, our approach leads to substantial savings in the search, since we only introduce the complicated models if necessary. It should be emphasized that the more complicated CMFs will involve more quantities that will be put on the stack and will therefore result in a much larger scenario model. Finally, we note that Nayak's methods for model simplification can be applied to the simulation model generated at every state from our scenario model.

Falkenhainer and Forbus' work on compositional modeling [7] describes the representation aspects of compositional modeling and addresses the model construction problem. In their framework, every model fragment has a set of *relevance conditions* corresponding to our modeling conditions. Our use of relevance claims enriches their language (specifically their *Consider* predicate) and provides it with a formal basis. In their model formulation algorithm, they first select the physical scope of the model (by identifying the lowest object down the partonomic hierarchy that subsumes all the objects mentioned in the query) and then select the relevant properties of these objects. They rely on heuristics to select types of properties to be modeled. This approach can easily lead to inclusion of model fragments that are not causally related to the query, and it cannot guarantee the sufficiency of the model produced. Our algorithm provides more flexibility in that the selection of the physical scope of the scenario model and the selection of the relevant properties are done in a uniform way, (by reasoning about the modeling constraints) and can therefore affect each other. Furthermore, we only select properties to model that can casually influence the query. Finally, to select the simplest model, they generate all possible consistent sets of modeling assumptions and choose the simplest, where simplicity is measured based on the number of objects included in each model and also on the ordering among choices within each assumption class.

While our model formulation algorithm may be more efficient than that of Falkenhainer and Forbus', especially under the assumptions listed in Section 5, their approach is more general than ours in some respects. In particular, Falkenhainer and Forbus do not require that each assumption class has a unique simplest and a unique most complex CMF, which we do. They also do not require one to represent formally the difference in the modeling assumptions among alternatives in an assumption class, which is likely to make the actual task of constructing a model fragment library much easier for their approach. This last point is an important consideration in practice because one needs to construct a large library of model fragments that can be used in a variety of situations if the compositional modeling paradigm is to be a viable option. The construction of a large model fragment library would certainly require collaboration among many domain experts. We have started an effort towards providing an environment for collaborative construction as well as use of a model fragment library [13]. It remains to be seen whether the type of organization we proposed here for a model fragment library with an explicit representation of modeling assumptions underlying each model fragment can be reliably constructed and maintained in such a collaborative environment.

Rickel and Porter's work on model formulation [27,28] is similar to ours since it makes use of graphs of interaction paths among quantities to select relevant model fragments. Their approach also provides guarantees of simplicity and adequacy. However, their graph of interactions is less general than the causal influence graph created by our algorithm, since it only includes variables, while we include all terms and predicates that could directly or indirectly influence the goal terms.

The idea of explicitly representing the differences between CMFs in an assumption class is similar to the graph of models by Addanki et al. [1]. Their work addresses the problem of selecting among complete models. Since the models in their graph are complete models instead of fragments, the space requirement of their approach increases exponentially as the number of possible modeling assumptions increases. Our approach can be viewed as combining the idea of a graph of models with compositional modeling.

The model simplification problem has been addressed by Williams [33] and Weld [32]. Williams also makes use of causal influence graphs to simplify a model. Both Weld and Williams assume a complete model of the situation as an input. Williams also makes use of the idea of following causal influences in his work on innovative design [34].

7. Discussion

This document describes a method for selecting model fragments to generate a scenario model that is appropriate for answering a given query. The method is based on a general framework for relevance reasoning, and we have argued that the modeling problem can significantly benefit from being considered from the perspective of relevance reasoning. Specifically, we have shown that some aspects of the modeling problem can be approached using general considerations of relevance reasoning, namely, backward chaining on causal influences and articulating the differences between CMFs in assumption classes. Moreover, we have shown how to incorporate engineering knowledge and heuristics for modeling in a declarative fashion, using relevance reasoning. The ability to declaratively express modeling heuristics has important advantages. Since it is easier to inspect and modify declarative knowledge, experimenting with different modeling heuristics becomes viable. Our approach offers a novel model formulation algorithm which efficiently selects a simplest model for a system and a query. An important aspect of our algorithm is that it chooses a model for a simulation of the system without knowing precisely what states the system can reach.

The algorithm has been implemented as part of a system called the Device Modeling Environment (DME) [14], which provides a computational environment for designing electro-mechanical devices. Given a topological description of a device, DME formulates a behavior model of the device using the compositional modeling approach and simulates its behavior. Prior to implementing our algorithm, the system would prompt the user to select a set of model fragments to be considered in the scenario model, thus creating a significant knowledge acquisition bottleneck. DME checks the operating conditions of every model fragment in the scenario model to determine the simulation model for each state. The system has been tested on many examples, including those used in this document.

Research on compositional modeling is in its infancy. The discussion in this document contributes by crystallizing some of the main questions regarding the approach that require additional research. The key issues that came to bear in this work are:

(1) How to write 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 a model fragment),
(2) How to organize model fragments in a library, and
(3) What assumptions can be made about the model fragment library.

We have contributed to solving problem (2) by suggesting the concept of composite model fragments and by requiring explicit representation of the differences between CMFs in an assumption class. In our discussion, we made several assumptions regarding questions (1) and (3). In particular, in order for our algorithm to produce a simplest model in polynomial time, we made the following assumptions:

(1) The model-fragment library is complete and coherent, and the scenario model consisting of all the possible instantiations of CMFs that are the most complicated in their assumption class is adequate for answering the query.
(2) The assumption classes are linear, and the simpler CMFs are causal approximations of the more complicated CMFs.
(3) There are no disjunctions in the consequences clause of a model fragment.
(4) Other modeling constraints are specified using only Horn rules.

In general, we see a tradeoff between (1) and (3). If more assumptions are made about individual model fragments, then fewer constraints need to be placed on the model library as a whole, and vice versa. Finding the optimal point in the spectrum of possibilities requires additional research and practical experience building systems. We believe that imposing some structure on the model fragment library is necessary and beneficial in the long run to facilitate knowledge acquisition and reuse.

The problem of model formulation can be viewed as one instance of a problem solving setting in which a system needs to reason *about* its own knowledge before answering a query. In doing so, it must choose among alternative representations of the domain that make different assumptions and abstractions. Other instances of this problem are also currently under investigation, such as reasoning with contexts and query evaluation in distributed heterogeneous databases [19]. We believe that the techniques developed in this work can form the basis for reasoning mechanisms in these other problem solving tasks.

Finally. the most serious weakness of our approach, from the practical point of view, is that it requires all the representational apparatus introduced in this article, in particular assumption classes as graphs of CMFs and links with explicit representations of underlying modeling assumptions. It is unarguably a very arduous task to construct such assumption classes. Since alternative model fragments about a phenomenon actually result from approximating or abstracting a detailed "most faithful" one in some way, gaining a better understanding of the nature of different approximation and abstraction techniques is one of our current goals. With such an understanding, we hope some day to be able to semi-automate the task of constructing a whole assumption class from the most detailed model fragment for each phenomenon.

Acknowledgement

The authors would like to thank Jeff Rickel and Pandu Nayak for helpful discussions on this article and the problem of model formulation in general. We also thank the anonymous reviewers of this article for detailed comments, which were very valuable in improving the quality of the final version. We would also like to thank the past and the current members of the How Things Work project at Stanford's Knowledge Systems Laboratory for their help in shaping the ideas in this paper and in the implementation of DME. Pandu Nayak provided the original knowledge base for the bimetalic strip example. Patrice Gautier constructed the original knowledge base for the reaction control system example. Diane Chi constructed the much enhanced version of the RCS example used for this article. This research was sponsored in part by the Defense Advanced Research Projects Agency (DARPA) and by NASA Ames Research Center, under NASA grants NAG 2-581 and NCC 2-537. It was also supported in part by DARPA and the National Institute for Standards and Technology, Rapid Development, Exploration and Optimization (RaDEO) program, under cooperative agreement 70NANB6H0075.

Appendix A. Proof of Theorem 8

Proof of termination. First, we note that because of assumption T4, the algorithm (i.e., the while loop in **select-scenario-model**) must terminate. This is because we can always adjust a choice of a CMF to a more complicated CMF that does not contradict the assumptions in *Rel*. Ultimately, we will end up with the most complicated scenario model, which is guaranteed, by T4, to be adequate.

Proof of adequacy. Let L be the logical model which satisfies the positive literals in *Rel* and the negation of positive literals not appearing in *Rel*. L is a logical model of $C \wedge Init$ because $Init \subseteq Rel$ and $Rel \cup C$ is closed under deduction.

Proof of the condition D1 *of adequacy*:

Condition D1.1 is satisfied because the following holds in the algorithm:

(1) For every CMF $c \in \mathcal{S}$, $Pos(As_c) \subseteq Rel$.
We always add to *Rel* $Pos(As_c)$ of any CMF c that is added to $\mathcal{S}$ (and if the choice of c is later adjusted to c', then $Pos(As_{c'}) \supseteq Pos(As_c)$ because of the causal approximations assumption, T2).

(2) Whenever some simplifying assumption of a CMF c is not satisfied in *Rel* (i.e., $\neg p \in As_c$ but $p \in Rel$), we adjust the choice of c.

Because of (1), all the positive literals in c are satisfied in L. Because of (2), all the negative literals of c are satisfied in L.

Condition D1.2 is satisfied by L because whenever $\texttt{Rel}(q_1) \in Rel$, where q_1 is a quantity, then either:

- $q_1 \in E$, or
- q_1 was put on the queue, and some assumption class that can affect q_1 was subsequently added to $\mathcal{S}$.

Therefore, in both cases, $\mathcal{S}$ will contain a CMF that includes q_1.

Proof of the condition D2 *of adequacy*:

In building the scenario model, we considered all the assumption classes that can affect q. The operating conditions of a CMF from at least one of these assumption classes must be in $\mathcal{S}_s$, since otherwise, there would be no model for the system in the state. Therefore $\mathcal{S}_s$ includes the quantity q. The library coherence assumption T1 guarantees that the set of equations in $\mathcal{S}_s$, which we denote by Eq_s, is not over-constrained.

If Eq_s is not over-constrained, the library completeness assumption guarantees that Eq_s can be made self-contained by choosing some quantities in the set $E_{input} \cup E_{global}$ to be exogenous in Eq_s. Suppose it cannot. This means that Eq_s can only be made self-contained by assuming some other quantities in Eq_s to be exogenous. Let x be one such quantity. Therefore, $x \notin E_{input} \cup E_{global}$ and q causally depends on x.

However, this contradicts the fact that we included all the assumption classes (therefore, all the equations) that can affect q in any situation and the assumption that the library is complete.

Proof of simplicity. In this proof, it is important to remember that we have only partial knowledge about the possible states that the system may reach. Specifically, all we know is that the time independent facts given in the system description must hold and that the value of binary quantities given in E_{input} cannot change.

In the proof, we assume that $\mathcal{S}$ was constructed by adding the CMFs c_i from assumption class a_i at the ith iteration of **select-from-assumption-class**. Note that some of the c_i's may have been removed subsequently by choosing a more complicated CMF from the same assumption class. We prove the following by induction on i:

A1 There must be a CMF in $\mathcal{S}$ from the assumption class a_i.

A2 The CMF c_i is the simplest CMF that can be chosen from a_i, with respect to $\mathcal{C}$.

A3 For each quantity q_1 on the queue, we must include all the phenomena that can affect q_1 and can occur in one of the possible states of the system.

Conditions A1 and A3 guarantee that all the phenomena modeled in $\mathcal{S}$ are necessary, A2 guarantees that all these phenomena are modeled in the simplest way possible with respect to the modeling constraints, $\mathcal{C}$. The simplicity of $\mathcal{S}$ follows from these claims.

The base case includes all the assumption classes that can affect the query quantity q. Clearly, A1 is satisfied because we need an assumption class that can determine q, and the ones that were chosen were those that are consistent with the possible states of the system. Since the assumption classes are linear and **select-from-assumption-class** selects the simplest CMFs in these assumption classes that do not contradict *Rel*, condition A2 is satisfied. Condition A3 is satisfied because if a quantity q_1 appears with q in the same equation, then q_1 can causally influence q. If q_1 is not exogenous, then any phenomenon that can influence q_1 must be included in the model. Similarly, if q_1 appears in the operating conditions of a CMF that can determine q and is not exogenous, then any phenomenon that can affect q_1 must be included in the model.

We assume the claims for i, and we prove them for $i+1$. The CMF c_{i+1} could have been added in two ways. In the first, we use the outer loop (i.e., adding a new assumption class when popping a quantity from the queue). By the inductive assumption, we must include all the phenomena that can affect the quantity on the top of the queue. Therefore, adding CMF from a_{i+1} is necessary, and so A1 is satisfied. As before, A2 and A3 are satisfied because **select-from-assumption-class** selects the simplest CMF c that satisfies the assumptions made so far and adds only the necessary quantities to the queue.

The second possibility for adding c_{i+1} is by the inner loop (i.e., by adjusting a previous choice from an assumption class). In this case, the inclusion of a CMF from a_{i+1} was justified by a previous CMF added to S. Since the modeling assumptions in *Rel* include only those that are entailed by $\mathcal{C}$ and previous modeling assumptions, they are therefore the minimal set of assumptions, and since c_{i+1} is the simplest CMF from a_{i+1} that can be included in the scenario model, A2 is therefore satisfied.[23] Finally, the quantities that were put on the queue when c_{i+1} is put in $\mathcal{S}$ are necessary using the same argument as before. Moreover, any quantity that is already on the queue does not have to be removed, because the CMF c_{i+1} is replacing a CMF c_j that is a causal approximation of c_{i+1}, and therefore, any causal influence that was possible through c_j will be possible through c_{i+1}. $\square$

References

[1] S. Addanki, R. Cremonini and J. Penberthy, Reasoning about assumptions in graphs of models, in: *Proceedings IJCAI-89*, Detroit, MI (Morgan Kaufmann, Los Altos, CA, 1989) 443–446.

[23] Note that if c_{i+1} was put in $\mathcal{S}$ instead of c_j for $j < i+1$, then c_j was removed from $\mathcal{S}$.

[2] D. Bobrow, B. Falkenhainer, A. Farquhar, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki and B. Kuipers, A compositional modeling language, in: *Qualitative Reasoning, Proceedings 10th International Workshop* (AAAI Press, Menlo Park, CA, 1996) 12–21; also: AAAI Tech. Rept. WS-96-01.

[3] D. Bobrow, B. Falkenhainer, A. Farquhar, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki and B. Kuipers, Cml: a compositional modeling language, Tech. Rept. KSL-94-16, Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford, CA (1994).

[4] R. Carnap, *Logical Foundations of Probability* (University of Chicago Press, Chicago, IL, 1950).

[5] J. Crawford, A. Farquhar and B. Kuipers, A compiler from physical models into qualitative differential equations, in: *Proceedings AAAI-90*, Boston, MA (Morgan Kaufmann, Los Altos, CA, 1990) 365–372.

[6] M.M. Denn, *Process Modeling* (Pitman, Marshfield, MA, 1986).

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

[8] K.D. Forbus, Qualitative process theory, *Artificial Intelligence* 24 (1984) 178–219.

[9] P. Gärdenfors, On the logic of relevance, *Synthese* 37 (1978) 351–367.

[10] M.R. Genesereth and R.E. Fikes, Knowledge interchange format, version 3.0 reference manual, Tech. Rept., Logic-92-1, Logic Group, Stanford University, Stanford, CA (1992).

[11] Y. Iwasaki and C.M. Low, Model generation and simulation of device behavior with continuous and discrete change, *Intelligent Systems Engineering* 1 (2) (1993) 115–145.

[12] Y. Iwasaki and H.A. Simon, Causality in device behavior, *Artificial Intelligence* 29 (1986) 3–32.

[13] Y. Iwasaki, A. Farquhar, R. Fikes and J. Rice, A web-based compositional modeling system for sharing of physical knowledge, in: *Proceedings IJCAI-97*, Nagoya, Japan (Morgan Kaufmann, San Francisco, CA, 1997).

[14] Y. Iwasaki and C.M. Low, Device modeling environment: an integrated model-formulation and simulation environment for continuous and discrete phenomena, in: *Proceedings Conference on Intelligent Systems Engineering* (The Institution of Electrical Engineers, London, UK, 1992) 141–146.

[15] J.M. Keynes, *A Treatise on Probability* (Macmillan, London, 1921).

[16] A.Y. Levy, Irrelevance reasoning in knowledge base systems, Ph.D. Thesis, Stanford University, Stanford, CA (1993).

[17] A.Y. Levy, Creating abstractions using relevance reasoning, in: *Proceedings AAAI-94*, Seattle, WA (AAAI Press/MIT Press, Menlo Park, CA, 1994) 588–594.

[18] A.Y. Levy, R.E. Fikes and S. Sagiv, Speeding up inferences using relevance reasoning: a formalism and algorithms, *Artificial Intelligence* 97 (1997) 83–136.

[19] A.Y. Levy, A. Rajaraman and J.J. Ordille, Query answering algorithms for information agents, in: *Proceedings AAAI-96*, Portland, OR (AAAI Press/MIT Press, Menlo Park, CA, 1996) 40–47.

[20] A.Y. Levy and Y. Sagiv, Constraints and redundancy in datalog, in: *Proceedings 11th ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems*, San Diego, CA (ACM, New York, NY, 1992) 67–80.

[21] A.Y. Levy and Y. Sagiv, Exploiting irrelevance reasoning to guide problem solving, in: *Proceedings IJCAI-93*, Chambery, France (Morgan Kaufmann, San Mateo, CA, 1993) 138–144.

[22] D. Macaulay, *The Way Things Work* (Houghton Mifflin, Boston, MA, 1988).

[23] P.P. Nayak, Automated model selection, Ph.D. Thesis, Stanford University, Stanford, CA (1992).

[24] P.P. Nayak, Causal approximations, *Artificial Intelligence* 70 (1994) 277–334.

[25] P.P. Nayak and L. Joskowicz, Efficient compositional modeling for generating causal explanations, *Artificial Intelligence* 83 (1996) 193–227.

[26] J. Rickel and B. Porter, Automated modeling for answering prediction questions: Exploiting interaction paths, in: *Proceedings 6th International Workshop on Qualitative Reasoning about Physical Systems*, Edinburgh, Scotland (1992) 82–95.

[27] J. Rickel and B. Porter, Automated modeling for answering prediction questions: Selecting the time scale and system boundary, in: *Proceedings AAAI-94*, Seattle, WA (AAAI Press/MIT Press, Menlo Park, CA, 1994) 1191–1198.

[28] J. Rickel and B. Porter, Automated modeling of complex systems for answering prediction questions, *Artificial Intelligence* 93 (1997) 201–260.

[29] H.A. Simon and N. Rescher, Cause and counterfactual, *Philos. Sci.* 33 (1966) 323–340.

[30] D. Subramanian and M.R. Genesereth, The relevance of irrelevance, in: *Proceedings IJCAI-87*, Milan, Italy (Morgan Kaufmann, Los Altos, CA, 1987) 416–422.

[31] D. Subramanian, A theory of justified reformulations, Ph.D. Thesis, Stanford University, Stanford, CA (1989).

[32] D.S. Weld, Approximation reformulation, in: *Proceedings AAAI-90*, Boston, MA (AAAI Press/MIT Press, Menlo Park, CA, 1990) 407–412.

[33] B.C. Williams, Capturing how things work: constructing critical abstractions of local interactions, in: *Proceedings Workshop on Automatic Generation of Approximations and Abstractions* (1990) 163–174.

[34] B.C. Williams, Interaction-based invention: designing novel devices from first principles, in: *Proceedings AAAI-90*, Boston, MA (AAAI Press/MIT Press, Menlo Park, CA, 1990) 349–356.