Automated Model Selection for Simulation

Yumi Iwasaki
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C
Palo Alto, California, 94304
iwasaki@cs.stanford.edu

Alon Y. Levy
AT&T Bell Laboratories
AI Principles Research Department
600 Mountain Ave., Room 2C-406
Murray Hill, NJ 07974
levy@research.att.com

Abstract

Constructing an appropriate model is crucial in reasoning successfully about the behavior of a physical situation to answer a query. In compositional modeling, a system is provided with a library of composible pieces of knowledge about the physical world called model fragments. Its task is to select appropriate model fragments to describe the situation, either for static analysis of a single state, or for the more complicated case simulation of dynamic behavior over a sequence of states. In previous work we showed how the model construction problem in general can advantageously be formulated as a problem of reasoning about *relevance*. This paper presents an actual algorithm, based on relevance reasoning, for selecting model fragments efficiently for the case of simulation. We show that the algorithm produces an adequate model for a given query and moreover, it is the simplest one given the constraints in the query.

Introduction

Constructing an appropriate model is crucial in reasoning successfully about the behavior of a physical situation to answer a query. In the compositional modeling approach [Falkenhainer and Forbus, 1991], a system is provided with a library of composible 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. This paper presents a general algorithm for model construction for 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.

The intuition underlying our algorithm is that the model construction problem can be viewed as a problem of relevance reasoning [Subramanian and Genesereth, 1987; Levy and Sagiv, 1993]. Two types of relevance reasoning occur in this context. First, relevance reasoning is used to determine which phenomena can affect the query. Intuitively, assuming that the goal of modeling is to explain how the value of a term changes over time, what is relevant to this goal is all the things that could causally influence the term. Consequently, the high level mechanism driving the algorithm is backward chaining on the possible causal influences on the goal term, determining the set of phenomena, terms and objects that can affect the goal term directly or indirectly.

The second aspect of relevance reasoning is deciding which level of detail is relevant to the goal. Often there are multiple model-fragments describing the same phenomenon, (grouped into *assumption classes* [Falkenhainer and Forbus, 1991]). Choosing between them depends on the underlying modeling assumptions being made, i.e., on the abstractions of the domain and the approximations being made. Deciding to make a certain abstraction can be viewed as stating that some detail is irrelevant to the goal and can hence be removed from the representation of a phenomenon (see [Levy, 1994] for a more detailed account of the connection between irrelevance and abstractions). We therefore use relevance reasoning to decide which modeling assumptions are appropriate for the goal. Furthermore, explicit relevance claims can be used to express additional domain knowledge that comes to bear in selecting a model, thereby incorporating such knowledge in a principled manner. Our algorithm alternates between the two kinds of relevance reasoning. Given some modeling assumptions it determines which additional phenomena are relevant to the goal, and given a set of relevant phenomena it chooses the level of detail at which to model them. The algorithm can be shown to run in time polynomial in the size of the resulting

model under certain reasonable assumptions.

Compositional modeling is a very powerful approach to automated modeling of physical devices. However, to enable efficient model construction, we must enforce additional structure on the model fragment library and the model-fragments (e.g., [Nayak, 1992b]). An important contribution of this paper is identifying additional structure that can be imposed on a model fragment library that is both natural and facilitates efficient model construction.

Several pieces of work have addressed the model formulation problem for the compositional modeling approach [Falkenhainer and Forbus, 1991; Nayak, 1992a; Rickel and Porter, 1994]. Our work is distinguished in that it derives its generality from general considerations of relevance reasoning. Specifically, it combines model formulation for simulation with guarantees of adequacy and simplicity, which are not found in other works.

Knowledge representation and behavior prediction

Before describing the model formulation problem, we briefly describe the representation of physical knowledge and the simulation method based on the compositional model approach. Compositional modeling was first described by Forbus in his work on Qualitative Process Theory (QPT) [Forbus, 1984], and is also the basis of subsequent works on qualitative modeling by Falkenhainer and Forbus [Falkenhainer and Forbus, 1991], Crawford and Farquhar [Crawford *et al.*, 1990] and Iwasaki and Low [Iwasaki and Low, 1993]. In compositional modeling, a physical situation is modeled as a collection of model fragments. Each fragment represents some aspect of a physical object or a physical phenomenon. A model fragment consists of conditions and consequences. The condition part specifies the conditions under which the phenomenon occurs, including the individuals that must exist and the conditions they must satisfy for the phenomenon to occur. The consequences specify the functional relations among the attributes of the objects that are entailed by the phenomenon.

If there exists individuals $a_1, \ldots, a_n$ that satisfy the conditions of a model fragment M at time t, we say that an instance of M is active at that time. We will denote the instance as $M(a_1, \ldots, a_n)$ and call $a_1, \ldots, a_n$ its *participants*. The existence of an active model fragment instance implies that the variables mentioned in it are defined and that the consequences hold.

The basic idea behind the prediction mechanism is the following: for a given situation, the system identifies active model fragment instances by evaluating their conditions. We will call the set of active model fragments in each state the *simulation model*. The simulation model gives rise to equations that must hold among variables as a consequence of the phenomena taking place. The equations are used to determine the next state of the situation. Each state has a simulation model along with a set of variable values and predicates that hold in the state. In order to perform simulation effectively, we must be able to efficiently select the appropriate set of model fragments at every state. The output of our algorithm is a small set of model fragment instances, from which the system selects a simulation model at every step of the simulation.

Problem definition

This section defines the model formulation problem as well as some key concepts in our approach. The following problem definition is based on that by Falkenhainer and Forbus ([Falkenhainer and Forbus, 1991], p. 98) with some modifications explained below:

Given a *scenario description*, a *domain theory*, and a *query* about the scenario's behavior, the model formulation problem is to produce the most useful, coherent *scenario model*. We elaborate on each part of this definition.

Scenario description: Our scenario description specifies a set of facts about the initial state of the situation to be modeled. This typically includes a set of individuals (i.e., components of the system), their properties and relations among them, representing the physical structure in the initial state of the simulation.

Domain theory: The domain theory is represented as a library of model fragments. As stated, each model fragment has a set of conditions which are further divided to *operating conditions* and *modeling assumptions*. The operating conditions are conditions on values of variables in a current state that are required for the model fragment to be applicable. The modeling assumptions are meta-level conditions describing the way we have decided to describe the domain, and are meant to accommodate different ways of modeling a certain phenomenon. Examples include assumptions about the temporal granularity of the model, the simplifying assumptions or approximations it makes, or the accuracy level it provides. While operating conditions of a model fragment in the chosen scenario model may be satisfied at a certain step of the simulation and cease to hold at a later point, modeling assumptions are assumed to hold throughout the simulation. We make the following assumptions about the library of model fragments. Their formal definitions are given in [Levy *et al.*, 1994].

Coherence of the Library: The *library coherence* assumption requires that if we have a set of model fragments that have consistent modeling assumptions and whose operating conditions are satisfied, then the resulting set of equations will not be over constrained (i.e., will not have more equations that quantities).

Completeness of the Library: The *library completeness* assumption requires that the library contains knowledge about all the phenomena that can causally

affect any term appearing in any model fragments in the library unless the term is explicitly known to be at the boundary of the library's knowledge. We will call the set of terms that are known to be at the boundary of knowledge contained in the library *L the globally exogenous terms*, $E_{global}(L)$. $E_{global}(L)$ is the set of all terms such that they appear in model fragments but for which the library may not contain model fragments of all the phenomena that can directly causally affect the term in all possible circumstances.

In addition to the basic representation of physical knowledge as model fragments, we have additional constructs on model fragments, *composite model fragments* and *assumption classes*. These constructs facilitate model formulation by introducing a higher organizational structure into the model library.

A composite model fragment (CMF) is a set of model fragments that represent behaviors of the same components or process under different operating regions. For example, the voltage produced by a rechargeable battery is a different function of its charge-level in three different ranges of the charge-level. This characteristic of a battery is represented by three model fragments with different conditions on the charge-level. However, they can be seen as forming one complete "description" of a particular aspect of the battery behavior over the entire range of its charge-level. All model fragments in the library are grouped together into such sets, though a model fragment may constitute a singleton CMF.

CMFs are further grouped into assumption classes. An assumption class is a set of CMFs that describe the same phenomenon based on different and contradictory modeling assumptions. Since CMFs in an assumption class are contradictory, at most one of them should be included in any scenario model.

Query: A query is expressed as a list of terms to be explained. We assume that the purpose of modeling is to explain how and why those terms change over time, i.e. to produce a causal account of how they change.

A query may also include an a priori list of explicit modeling assumptions. Such lists can be provided by the user in order to provide additional information about the kinds of explanation desired, such as the level of details. In particular, they can include a list of exogenous terms, i.e., terms whose values are determined by factors outside the scope of the current problem. They are also used to delimit the set of possible states that should be considered possible in the simulation.

Scenario model: Given the inputs described above, the model formulation problem is to generate a set of possible CMF instantiations (i.e., a list of pairs: CMF, participant list). During simulation, the operating conditions of those CMF instances will be evaluated in every state to generate a simulation model.

For a model formulation algorithm to be useful, a scenario model generated should be adequate yet as simple as possible. We say a model is adequate when it is consistent and is sufficient for answering the given query. A scenario model is said to be consistent if the union of all the modeling assumptions underlying its members is consistent. A scenario model is said to be sufficient for a given query, if it gives rise to a simulation model in every state that contains the complete causal paths from exogenous terms to the query term. We use the definitions of causal dependency relations given in [Vescovi *et al.*, 1993], which expands the notion of causal ordering among variables [Iwasaki and Simon, 1986] to include conditions on model fragments. Intuitively, a term t_i is directly causally dependent on another term t_j if the value of t_i is determined by that of t_j through an equation in which both t_i and t_j appear, or if t_j appears in the applicability condition of an equation that determines the value of t_i. Finally, a scenario model C_1 is said to be simpler than C_2, if for each CMF c_1 in C_1 there is some CMF c_2 in C_2 for which c_1 is simpler than c_2 or they are the same.

Model formulation algorithm

This section describes our model formulation algorithm in detail. Figure 1 shows the outline of the algorithm. Informally, the algorithm consists of making two choices. The first is deciding which assumption classes should be represented. The second is to decide which CMF should be included out of each of the assumption classes. The first choice is done by backward chaining through the possible causal influences on the goal terms. The second is made by reasoning about the modeling assumptions necessary to answer the query.

We explain each of the main steps in the loop: selection of assumption classes, selection a CMF out of each such assumption class, and deciding which terms to further backward chain on. We use the following example throughout.

> The example (see Figure 2) is a simple circuit containing a solar array (`SA1`) and a rechargeable battery (`BA1`). The figure shows the circuit, the scenario description, and the assumption classes in the domain theory. For each CMF in the domain theory, its consequences and the list of terms appearing in its operating conditions are shown. The query is `Voltage(BA1)`, with a list of exogenous terms, which includes all the terms mentioned in the scenario description except `Damaged(BA1)`. There is no a priori list of modeling assumptions.

Finding the Assumption Classes

As the goal of modeling is to explain how and why the goal terms change over time, the relevant things to include in a model are those that could causally influence the goal terms. The consequences of model fragments contain functional relations among quantities, specifying the ways quantities can influence each other.

We take the position that equations in model fragments describe the functional relations among the continuous variables involved in the modeled phenomenon

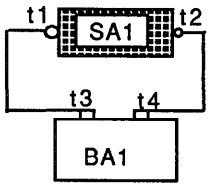
```
procedure select-scenario-model(v, E, Init, C)
/* v: the query variable */
/* E: the list of exogenous terms */
/* As_c: the modeling assumption of CMF c */
/* Q: a queue of terms */
/* Init: the list of modeling constraints in the query */
/* C: the background theory of modeling constraints */
/* Rel: the current list of modeling constraints */
/* Model is a list of pairs (c, x), where c is a potential instance of a CMF and x is a term c could causally affect */

begin
   Q = {v}.
   Rel = Init.
   Model = nil.
   repeat
      q = dequeue(Q).
      As = assumption classes in which q can be an output variable and
           whose operating conditions do not contradict E.
      for each a ∈ As do:
           select-from-assumption-class (a, q).
           while there is a pair (c, q') ∈ Model such that ¬p ∈ As_c and p ∈ Rel
                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 | (c, q') ∈ 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. */
/* Pos(As_c) is the list of positive literals in As_c.*/

begin
   c = The simplest CMF in A such that ∄ p(¬p ∈ As_c ∧ p ∈ Rel).
   Model = Model ∪ {(c, q)}.
   inputs = the union of:
      The quantities that appear in equations with q and
      The terms in the operating conditions of c.
   for every X ∈ inputs do
      if X has not been in Q and X ∉ E then
        enqueue X onto Q.
   Rel = DeductiveClosure(C ∪ Rel ∪ Pos(As_c)).
   if rel(q_1) ∈ Rel and q_1 ∉ E and q_1 has not been in Q then
      enqueue q_1 onto Q.
end select-from-assumption-class.
```

Figure 1: Model formulation algorithm

without specifying a particular causal direction. The direction of causality only emerges when the equation is embedded in a system of equations all representing independent mechanisms and the quantities that are externally determined are specified [Iwasaki and Simon, 1986]. In compositional modeling, this implies that the causal orientation of equations can only be determined after the model fragments are instantiated and equations are assembled into a simulation model. Therefore, one cannot a priori specify for each model fragment the quantity that is caused by the model fragment, but one can specify a priori 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. Given a term, the algorithm selects the assumption-classes that could affect it. In our implementation, we pre-compile a list of assumption classes that can affect each type of terms to facilitate this search.

For instance, in our example, a term of the form `Damaged(x)` where `x` is an instance of `Battery` can be causally influenced by a member CMF of `Battery-damage-due-to-overcharge-ac` when it is instantiated with `x` bound to `?b`.

The query term `Voltage(BA1)` is the only item



Scenario description

Solar-array(SA1)
Battery(BA1)
Rechargeable(BA1)
Plus-terminal(BA1) = t4
Minus-terminal(BA1) = t3
Plus-terminal(SA1) = t2
Minus-terminal(SA1) = t1
Electrically-connected(t2, t4)
Electrically-connected(t1, t3)
~Damaged(BA1).

Legend
CL: Charge-level(?b)
V : Voltage-produced(?b)
TEMP: Temperature-of(?b)
I: Current(Plus-terminal(?b))
DOD: Average-depth-of-discharge(?b)
TSLC: Time-since-last-conditioning(?b)
TPOG: Time-period-of-goal

Domain theory

CMF	Consequences	Terms in the operating conditions
Assumption-class: Battery-voltage-ac		
Constant-voltage-CMF	$V = c0$	Battery(?b), Damaged(?b)
Binary-voltage-CMF	$V = \begin{cases} 0 \text{ if } CL < c0 \\ v1 \text{ if } c0 \leq CL \end{cases}$	Battery(?b), Damaged(?b)
Normal-degrading-CMF	V = f(TIME)	Battery(?b), Damaged(?b)
Charge-sensitive-CMF	V = f(CL)	Battery(?b), Damaged(?b), Rechargeable(?b)
Temperature-sensitive-CMF	V= f(TEMP, CL)	Battery(?b), Damaged(?b), Rechargeable(?b)
Assumption class: Battery-charge-level-ac		
Constant-charge-level-CMF	$CL = c_1$	Battery(?b), Damaged(?b)
Normal-accumulation-CMF	$CL = \int Idt$	Battery(?b), Damaged(?b), Rechargeable(?b)
Accumulation-with-aging-CMF	$CL = \int Idt$ - f(DOD, TSLC)	Battery(?b), Damaged(?b), Rechargeable(?b)
Assumption-class: Battery-damage-due-to-overcharge-ac		
Battery-damage-CMF	Damaged(?b)	Battery(?b), Damaged(?b), Rechargeable(?b), CL(?b)

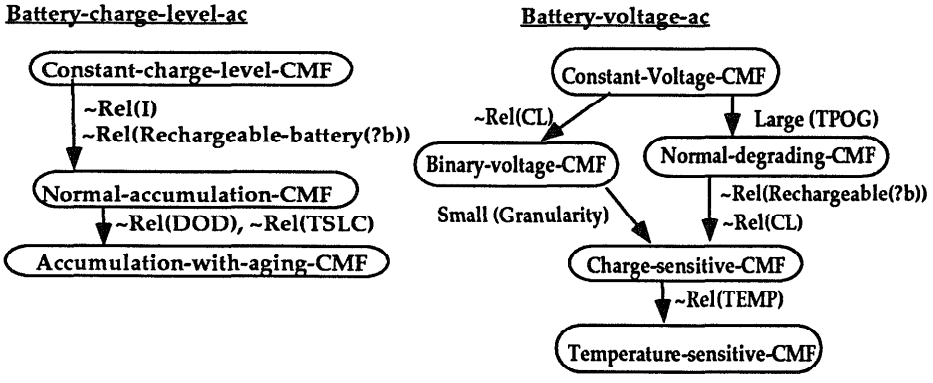


Figure 2: Model selection Example

on the queue initially, and it becomes the current goal.

`Battery-voltage-ac(BA1)` possibly has influence on the term, as `Voltage(BA1)` appears in the consequences of its member model fragments. Thus, we select this assumption class.

Selection of CMF out of an assumption class

Since CMFs in an assumption class represent alternative ways to model the same phenomenon, we must pick one CMF out of each assumption class thus deemed relevant. We make the choice by reasoning about the modeling assumptions being made about the problem. To facilitate the selection, we add to the representation of assumption classes additional structure that makes explicit the modeling assumptions that change between one CMF and another in the assumption class. Each assumption class is represented as a directed graph of CMFs. There is a link from a CMF c_1 to a CMF c_2 if c_1 is *simpler* than c_2.

The link is annotated with the difference in the modeling assumptions, i.e., the modeling assumptions that can be removed as one goes from c_1 to c_2. We employ the following convention in interpreting predicates in the modeling assumptions. A positive literal of a predicate denotes an assumption that yields a more complicated model of a phenomenon. A negative literal denotes a *simplifying* assumption. For example, the literal $\neg rel(Temperature(battery))$ in the modeling conditions of a model-fragment states that the fragment uses a simplified representation in which the temperature aspect of the battery is ignored, whereas the literal $rel(Temperature(battery))$ in the modeling con-

ditions states that the model fragment considers the temperature aspect of the battery. Not all modeling assumptions are relevance statements. An example of a modeling assumption that is not a relevance statement, denoted by the literal $\neg large(timeScale)$, states that the representation is simplified to ignore longer term effects on the battery.[1] A simplest CMF is one in which the least number of positive literals are entailed by the modeling assumptions. Finally, we assume that every assumption class has single simplest CMF, and a single most complicated CMF.

The graphs of CMFs in the assumption classes that have more than one CMF for our example are shown in Figure 2. In the assumption class `Battery-voltage-ac`, the CMF `Constant-voltage-CMF` assumes that the charge-level of the battery is irrelevant, while `Binary-voltage-CMF` does not.

Given this representation of assumption classes, we can choose a CMF by selecting the simplest one which does not contradict the modeling assumptions collected so far. After choosing the CMF, we update the modeling assumptions to include those implied by the chosen CMF (which, in particular, include the assumption that all the terms mentioned in that CMF are relevant). As a result of adding the new modeling assumptions, earlier choices of CMFs might be invalidated, since they may have been chosen based on stronger assumptions. In such cases, we adjust earlier choices by selecting more complicated model fragments out of the assumption classes from which they were chosen. Importantly, the number of times we will perform such adjustments is limited, and therefore, the complexity of our algorithm is not affected by these adjustments.

Returning to our example, we have just selected `Battery-voltage-ac`. To select a CMF out of this assumption class, we start from the simplest, `Constant-voltage-CMF`. Since there is no earlier relevance assumptions made so far, this choice is consistent, and we select this CMF. This results in addition of the following to our modeling assumption list:
`Rel(Battery(BA1))`,
`Rel(Damaged(BA1))`,
`Large(TPOG)`,
`¬Rel(Charge-level(BA1))`,
`Small(Granularity)`,
`¬Rel(Rechargeable(BA1))` and
`¬Rel(Temperature-of(BA1))`.

[1]Some assumptions may be multi-valued and the algorithms we describe in this paper can be extended in a straightforward fashion to deal with such assumptions. However, for clarity we assume here that modeling assumptions are binary.

Traversal of causal influence paths

Once a CMF is selected, we need to determine which causal paths to pursue further by deciding which terms to put on the queue. Essentially, we pursue the terms that can affect the goal term through the CMF. These are either terms that are part of the operating condition or terms that appear in the same equation as the goal in the consequence equations (and can therefore influence it in some causal ordering of the equations). We consider every such candidate term. If it is neither exogenous nor has already been put on the queue, it is pushed onto the queue. The procedure then calls itself recursively with the new queue.

Since `Damaged(BA1)` can influence `Voltage(BA1)` through `Constant-voltage-CMF(BA1)` and is not exogenous, it is placed on the queue and becomes the new goal.

A search through the model fragment library finds `Battery-damage-due-to-overcharge-ac` to influence the goal term. `Battery-damage-CMF` is selected from the assumption class since it is the only member.

This selection adds `Rel(Rechargeable(BA1))` and `Rel(Charge-level(BA1))` to the assumption list, making the assumption list inconsistent since both `¬Rel(Rechargeable(BA1))` and `¬Rel(Charge-level(BA1))` are included.

To resolve the inconsistency, we adjust the earlier choice of CMF from the assumption class, `Battery-voltage(BA1)`, since it resulted in the addition of these irrelevance assumptions. We now select `Charge-sensitive-CMF`, which is the simplest CMF that does not contradict the current modeling assumptions.

The goal term now becomes `Charge-level(BA1)`. A search through the model fragment library finds `Battery-charge-level-ac` to affect the term. The simplest CMF that is consistent with the current modeling assumptions in this assumption class is `Normal-accumulation-CMF(BA1)`, which we select. `Current(Plus-terminal(BA1))` can influence `Charge-level(BA1)` through this CMF. However, since it is an exogenous term, it is not placed on the queue. The queue is now empty and the procedure terminates.

The scenario model contains
`Charge-sensitive-CMF(BA1)`,
`Battery-damage-due-to-overcharge-CMF(BA1)`,
and `Normal-accumulation-CMF(BA1)`.

Analysis

This section describes the properties of the algorithm we presented. We first show that it produces an adequate model. We then explain under what conditions it produces the simplest model. Finally, we discuss the complexity of the algorithm. The complete proofs

are given in [Levy *et al.*, 1994]. In our discussion we assume (1) that all model fragments in a single CMF can determine the same set of variables and (2) that for each pair of CMFs, CMF_1 and CMF_2 in an assumption class such that CMF_1 is simpler than CMF_2, CMF_1 is a causal approximation [Nayak, 1992b] of CMF_2.

The second assumption is only necessary in order to assure that the algorithm as described will run in polynomial time. The algorithm can be modified to relax that assumption, but the resulting algorithm may not run in polynomial time. Nayak [Nayak, 1992b] shows that causal approximations capture most ones encountered in practice. Under these assumptions, our algorithm is guaranteed to produce an adequate and simplest model for the query, as stated by the following theorem.

Theorem 1: *Let $\mathcal{M}$ be a library of model fragments describing the domain, and C be a set of modeling constraints. Let S be a description of a system and $(v, E, Init)$ be a query about the system. Let $\mathcal{S}$ be the scenario model resulting from algorithm* **select-scenario-model**. *Furthermore, assume that:*

- *The library coherence assumption holds.*
- *If c_i and c_j are two CMFs in an assumption class, such that $c_i < c_j$, then c_i is a causal approximation of c_j.*
- *All modeling constraints in C are either ground atomic formulas or Horn rules.*
- *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.*[2]

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

The observation underlying the proof of adequacy is the following. Consider the graph of causal influences created by the algorithm. It consists of OR nodes (the goal nodes) and AND nodes (the CMF nodes). At every given state, the actual model used for that state is one of its subgraphs, in which every OR nodes has at most one arc emanating from it (i.e., for each goal, we choose at most one CMF that explains it). Consequently, since the graph represents possible paths from the exogenous variables to the goal, each of the subgraphs will too.

The algorithm also produces the simplest scenario model in the following sense. Recall that we are choosing the scenario model without performing the actual simulation. Consequently, we do not know which states the simulation will go through, and therefore which model-fragments will be required for those specific states. Instead, we assume that the simulation can go through *any* state that satisfies the input conditions, and our choice of a scenario model is made to accommodate any such state. The proof of simplicity is based on the correspondence between states of the simulation and subgraphs of the graph created by the algorithm.

The running time of the algorithm is polynomial in the number of CMF's in the scenario model. To see this, observe that although the algorithm sometime requires adjustments of previous choices, the maximum number of such adjustments is polynomial. Specifically, if d is the maximum number of CMFs in an assumption class, and n is the number of CMFs in the scenario model, the total number of adjustment steps is at most nd. This is because every adjustment step results in replacing some CMF with another that is more complicated. Because of the assumption that the simpler-than relation between a pair of CMFs in an assumption class is always a causal approximation, there is no possibility that a CMF can be replaced by a simpler CMF in the same assumption class without making the set of modeling assumptions inconsistent. After nd such replacements we will get the most complicated scenario model, which is guaranteed to be adequate.

Conclusions

We presented a novel algorithm that produces an adequate and simplest scenario model for simulation. The algorithm has three distinguishing aspects that are based on applying general considerations of relevance reasoning. The first is the backward chaining through causal influences, motivated by a general definition of relevance. The second is choosing the simplest possible CMF at each choice point, based on knowledge expressed as relevance claims. The third is that it reasons with partial knowledge of the states that might occur in the simulation.

We have implemented the algorithm as part of a system called, Device Modeling Environment (DME) [Iwasaki and Low, 1993], which is a device modeling program to provide a computational environment for design of electromechanical devices. Given the topological description of a device, DME formulates a model and simulates its behavior. The system works on several examples, including the electrical power system, of which the example used in Section 3 is a much simplified version.

Several researchers have proposed methods for model formulation. These works address one or both of the two aspects of model formulation problem, namely model construction and simplification.

Nayak [Nayak, 1992a] addressed both aspects of the problem in the context of the single state analysis. His algorithm for constructing a model also follows possible causal influences, but these influences must be given

[2] 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 definition of the model fragment and the time invariant facts in the description of the system.

explicitly using the *component interaction heuristic.* In contrast, our work addresses the problem in the more general context of dynamic behavior simulation, where the governing set of equations can change from state to state. Our approach exploits the structure of the model fragments to automatically derive the links that are given as component interaction heuristics in his approach. In choosing a model fragment from every assumption class, Nayak chooses the most complicated one, and later simplifies the resulting model. On the other hand, we build the model by selecting the simplest CMF possible in every class, adjusting the choice only if necessary. Therefore, the complexity of Nayak's algorithm is polynomial in the size of the *most complicated* model, while our is polynomial in the size of the *simplest* model. In complex systems, where the CMFs in an assumption class vary significantly in levels of detail, this will be a significant difference, and therefore our approach will be more practical.

Falkenhainer and Forbus [Falkenhainer and Forbus, 1991] 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. They rely on heuristics to select properties to be modeled. This approach can lead to inclusion of model fragments that are not causally related to the query, and cannot guarantee the sufficiency of the model. They attempt to produce the simplest model by generating all possible consistent sets of modeling assumptions and choosing the simplest based on an informal criteria of simplicity.

Rickel's work on model formulation is similar to ours since it makes use of graphs of interactions paths among quantities to select relevant model fragments [Rickel and Porter, 1994]. His graph of interactions are less general than our causal influence graph since it only includes quantities while we include all terms (including quantities, predicates, relations) that could directly or indirectly influence the goal terms. His approach also does not provide guarantees of sufficiency or simplicity.

The idea of graph of CMFs is similar to graph of models by Addanki et al. [Addanki *et al.*, 1989] for selecting among complete models. Since their models are complete models instead of fragments, the space requirement of their approach would increase exponentially as the number of possible modeling assumptions increases.

Acknowledgments

The authors would like to thank Richard Fikes and Pandu Nayak for many useful discussions. This research was sponsored by the Defense Advanced Research Projects Agency and by NASA Ames Research Center, under NASA grants NAG 2-581 and NCC 2-537.

References

Addanki, Sanjaya; Cremonini, R.; and Penberthy, J. 1989. Reasoning about assumptions in graphs of models. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.*

Crawford, James; Farquhar, Adam; and Kuipers, Ben 1990. A compiler from physical models into qualitative differential equations. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann.

Falkenhainer, Brian and Forbus, Ken 1991. Compositional modeling: Finding the right model for the job. In *Artificial Intelligence.* Vol. 51, pp. 95–143.

Forbus, Ken 1984. Qualitative process theory. In *Artificial Intelligence*, volume 24.

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

Iwasaki, Y. and Simon, H. A. 1986. Causality in device behavior. In *Artificial Intelligence.* Vol. 29.

Levy, Alon Y. and Sagiv, Yehoshua 1993. Exploiting irrelevance reasoning to guide problem solving. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence.*

Levy, Alon Y.; Iwasaki, Yumi; and Fikes, Richard 1994. Automated model selection for simulation based on relevance reasoning. Knowledge System Laboratory Technical Report. Computer Science Department, Stanford University. In preparation.

Levy, Alon Y. 1994. Creating abstractions using relevance reasoning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.*

Nayak, P. Pandurang 1992a. *Automated Model Selection.* Ph.D. Dissertation, Stanford University, Stanford, CA.

Nayak, Pandurang 1992b. Causal approximations. In *Proceedings of the Tenth National Conference on Artificial Intelligence.*

Rickel, Jeff and Porter, Bruce 1994. Automated modeling for answering prediction questions: Selecting the time scale and system boundary. In *Proceedings of the Twelfth National Conference on Artificial Intelligence.*

Subramanian, D. and Genesereth, M.R. 1987. The relevance of irrelevance. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann.

Vescovi, Marcos; Iwasaki, Yumi; Fikes, Richard; and Chandrasekaran, B. 1993. Cfrl: A language for specifying the causal functionality of engineered devices. In *Proceedings of the Eleventh National Conference on Artificial Intelligence.*