Automated Model Selection for Simulation

Yumi Iwasaki and Alon Y. Levy
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C
Palo Alto, California, 94304
({iwasaki,levy}@cs.stanford.edu)

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.

1 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. In our previous work [Levy *et al.*, 1992], we showed how 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. In doing so, we used a general framework for reasoning about relevance described in [Levy and Sagiv, 1993, Levy, 1993]. In this paper, we present an actual algorithm that is based on the 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.

Our algorithm has three key features based on different aspects of relevance reasoning. First, the framework provides a well motivated definition of the notion of relevance. 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 all the possible causal influences on the goal term. This allows the system to determine the set of all the things, including objects and physical phenomena, that could causally influence the goal term directly or indirectly.

Backward chaining is not sufficient for choosing a model. Often there are multiple ways to describe the same phenomenon, and these are grouped into *assumption classes* [Falkenhainer and Forbus, 1991]. Choos-

ing between them depends on the underlying modeling assumptions being made. As we showed in [Levy *et al.*, 1992], many of these modeling assumptions can be stated as *relevance claims*, in the language provided by the framework. Relevance claims enable the user to express additional domain knowledge that comes to bear in selecting a model. In the algorithm, we use relevance claims to guide in selecting models from assumption classes.

Finally, an important aspect of effective relevance reasoning is the ability to reason with partial knowledge, i.e., the ability to decide that certain facts are irrelevant to a problem even if we have partial knowledge about these facts or partial knowledge about the problem at hand. Formulating a model for simulation has exactly this flavor, since we don't know what states of the world need to be modeled a priori. Even given complete knowledge of the initial state, the future states are not known until we construct a model and perform simulation. Thus, to build a model for simulation, we must somehow determine what could be relevant to answering the query without performing simulation. Our algorithm chooses a model for simulation based only on knowledge of some constraints on the possible states, but without actually generating them.

Several pieces of work have addressed the model formulation problem for the compositional modeling approach [Falkenhainer and Forbus, 1991, Nayak, 1992a, Rickel and Porter, 1992]. Our work is distinguished in that 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. Section 5 provides further details.

1.1 Knowledge representation and behavior prediction

Before describing the model formulation problem, we briefly describe the representation of physical knowledge and the simulation method. 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, at time t, there exists individuals $a_1, \ldots, a_n$ that satisfy the operating conditions of a model fragment M we say that an instance of M is active at that time. We will call $a_1, \ldots, a_n$ the *participants* of the instance of M. We will also denote the particular instance by $M(a_1, \ldots, a_n)$.

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.

2 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 of 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 operating conditions which are further divided to *activation conditions* and *modeling assumptions*. The activation 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. For example, these include assump-

tions about the temporal granularity of the model, the simplifying assumptions or approximations it makes, or the accuracy level it provides. To contrast the two sets of conditions, activation conditions of a model fragment 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 (or not hold during all of it). We make two assumptions about the library of model fragments:

Consistency: Given an internally consistent set of modeling assumptions and a given state, the subset of the library whose conditions are satisfied forms a logically and mathematically consistent simulation model.

Completeness: Given a query, there is at least one consistent set of modeling assumptions such that the subset of the library based on the assumptions gives rise to a simulation model that can determine the value of the goal term in every state.

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 the term (or a list of terms) to be explained. A term may be quantitative such as voltage or a non-quantitative such as the connectivity between two objects. We assume that the purpose of modeling is to explain how and why those terms change over time; in other words, the purpose is to produce a causal account of the way those terms change over time.

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.

In contrast to the formulation given by [Falkenhainer and Forbus, 1991], we define the problem as that of selecting CMFs, not model fragments.

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. The exact definition of a causal relation between values of any terms in simulation states are given in [Iwasaki and Chandrasekaran, 1992]. 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.

3 Model formulation algorithm

This section describes our model formulation algorithm in detail. 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. The algorithm is outlined in Figure 1. 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.

3.1 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 specify a functional relation among quantities, and thus the ways one quantity can influence another. A term can be influenced by a model fragment only if it appears in its consequence.[1]

Therefore, for each type of term, we can compile a list of assumption classes which could possibly influence the term. This information is represented in our knowledge base in the *Can-affect-list*.[2] Given a term, the model formulation procedure looks up the Can-affect-list for assumption-classes that could affect the term, and returns a list of assumption classes.

> The Can-affect-list of our example is shown in Figure 2. For instance, 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 on the queue initially, and it becomes the current goal. Searching through the Can-affect-list, we identify `Battery-voltage-ac(BA1)` as possibly having influence on the term. Thus, we select this assumption class.

3.2 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 make 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. A CMF c_1 is said to be simpler than c_2 if it makes more modeling assumptions than c_2. The link is annotated with difference in the modeling assumptions, i.e., the modeling assumptions that can be removed as one goes from c_1 to c_2. We assume that every assumption class has single simplest CMF, and a single most complicated CMF.[3]

> The graphs of CMFs in the assumption classes that have more than one CMF for our example are shown in the figure. 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.

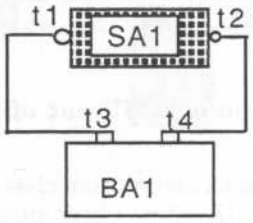
Given this representation of assumption classes, we can choose a CMF by selecting the simplest one which does not contradict the modeling assumptions collected so far. After we choose the CMF, we update the modeling assumptions to include those implied by the chosen CMF (which, in particular, include the assumption that we model all the terms mentioned in that CMF). 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

[1] Since we do not assume that the functional relations appearing as the consequence of model fragments are causally directed, quantities appearing in such relations could possibly influence each other in any direction. In general, it is only after the functional relation is placed in some context and exogenous quantities are specified that one can determine the direction of causal dependence among them.

[2] Note that this list can be generated automatically, however, in some cases it is better to do so manually in order to eliminate some causal influences that we don't want to be pursued.

[3] Note that this assumption is also made in other works, such as [Nayak, 1992b].



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)

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 = { 0 if CL < c0; v1 if c0 ≤ CL	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 = c1	Battery(?b), Damaged(?b)
Normal-accumulation-CMF	CL = Idt	Battery(?b), Damaged(?b), Rechargeable(?b)
Accumulation-with-aging-CMF	CL = 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)

Can-affect-list

Relation-name	Argument-types	Assumption-class
Voltage	(Battery)	Battery-voltage-ac(?b)
Charge-level	(Battery)	Battery-charge-level-ac(?b)
Damaed	(Battery)	Battery-damage-due-to-overcharge-ac(?b)

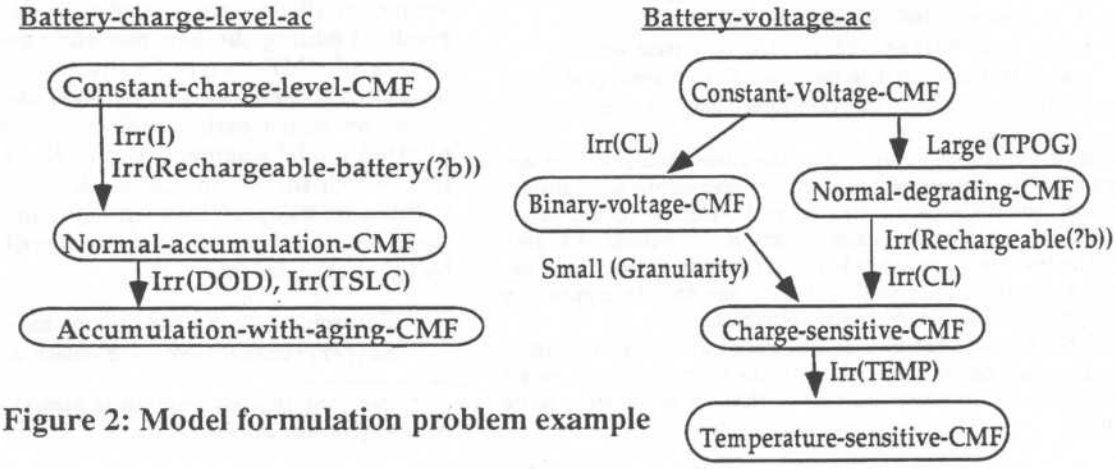


Figure 2: Model formulation problem example

Figure 2: Model selection Example

```
procedure select-scenario-model(query)
begin
    Q = goal terms of query
    Assumptions = modeling assumptions given in the query.
    Model = nil.
    repeat
        q = dequeue(Q)
        As = assumption classes that can influence q
        for each a ∈ As do:
            Select the simplest CMF in a that does not contradict Assumptions and add it to Model.
            Update Assumptions.
            Adjust previous CMF choices.
            for all terms x such that x can influence g through As do
                if x has been expanded and is not exogenous then
                    enqueue x onto Q.
    until Q is empty.
    Model is the scenario model.
end
```

Figure 1: Model selection algorithm.

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)`, `Irr(Charge-level(BA1))`, `Small(Granularity)`, `Irr(Rechargeable(BA1))` and `Irr(Temperature-of(BA1))`.

3.3 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 current goal.

A search through the Can-affect-list finds `Battery-damage-due-to-overcharge-ac`, out of which `Battery-damage-CMF` is selected since it is the only member.

This selection adds `Rel(Rechargeable(BA1))` and `Rel(Charge-level(BA1))` to the assumption list.

However, this makes the assumption list inconsistent since both `Irr(Rechargeable(BA1))` and `Irr(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 current goal term now becomes `Charge-level(BA1)`. A search through the Can-affect-list finds `Battery-charge-level-ac`. 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 resulting scenario model contains:
`Charge-sensitive-CMF(BA1)`,
`Battery-damage-due-to-overcharge-CMF(BA1)`,
`Normal-accumulation-CMF(BA1)`.

Notice that the procedure terminates here in this example because `Current(Plus-terminal(BA1))` was specified exogenous in the query. Had it not been specified exogenous, the procedure would

have included more CMFs including those representing other components and processes affecting the current.

4 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. Due to space limitations, we do not present the full proofs. These can be found in [Levy, 1993]. In our discussion we assume (1) that all model fragments in a single CMF can determine the same set of variables and (2) that All approximation relations are causal approximations [Nayak, 1992b].

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

Theorem 4.1: *Let C be the set of CMF's chosen by the algorithm for the query v. The set C is an adequate scenario model for v, i.e., at every given point in the simulation, there is a causal path from the exogenous values to the current value of v.*

The observation underlying the proof of the theorem 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. A scenario model C_1 is 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.

Theorem 4.2: *Let C be the scenario model chosen by the algorithm. Given that the simulation may go through any state that satisfies the input conditions, there is no scenario model C', such that C' is simpler than C and is an adequate model for the query.*

Here too, the proof 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 requires that we sometimes adjust previous choices of CMFs, the total number of times backtracking can be done 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 backtracking steps is at most nd. This is because every backtracking step results in replacing some CMF with anothes that is more complicated. After nd such replacements we will get the most complicated scenario model, which is guaranteed to be adequate.

5 Conclusions

We presented a novel algorithm for choosing a 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 model, 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. The algorithm is shown to produce an adequate and most simple model. Moreover, it is shown to be efficient.

We have implemented the algorithm as part of a system called, Device Modeling Environment (DME) [Iwasaki and Low, 1992], 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 using the compositional modeling approach 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 . 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 to further focus 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. Later, he uses a procedure to simplify the resulting model. Our work attempts to build the model by selecting the simplest CMF possible in every class, and only adjust the choice if necessary. In cases where the CMFs vary significantly in their complexity this will lead to substantial savings in the search, since in our approach we only introduce the complicated models if necessary. It should be emphasized that the more complicated CMFs will give rise to additional subgoals in the backward chaining, thereby significantly raising the branching factor the graph unnecessarily.

Forbus and Falkenhainer's work [Falkenhainer and Forbus, 1991] mainly addresses the construction problem. They 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 also 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. They attempt to produce the simplest model, by generating all possible consistent sets of modeling assumptions and choosing the simplest based on a very 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. 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]. 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 would increase exponentially as the number of possible modeling assumptions increase.

The model simplification problem has been addressed by Williams [Williams, 1990a] and Weld [Weld, 1990]. 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 makes use of the idea of following causal influences also in his work on innovative design [Williams, 1990b].

References

[Addanki *et al.*, 1989] 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.*

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

[Iwasaki and Chandrasekaran, 1992] Iwasaki, Yumi and Chandrasekaran, B. 1992. Design verification through function and behavior-oriented representations: Bridging the gap between function and behavior. In *Proceedings of the Second International Conference on Artificial Intelligence in Design.* Kluwer Academic Publishers.

[Iwasaki and Low, 1992] Iwasaki, Yumi and Low, Chee Meng 1992. Device modeling environment: An integrated model-formulation and simulation environment for continuous and discrete phenomena. In *Proceedings of Conference on Intelligent Systems Engineering.*

[Levy and Sagiv, 1993] 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 *et al.*, 1992] Levy, Alon; Iwasaki, Yumi; and Motoda, Hiroshi 1992. Using relevance reasoning to guide compositional modeling. In *The Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence, Seoul, South Korea. Also appears in the Proceedings of the Workshop on Approximations and Abstractions of Computational Theories, AAAI-92, San Jose, CA.*

[Levy, 1993] Levy, Alon Y. 1993. *Irrelevance Reasoning in Knowledge Base Systems.* Ph.D. Dissertation, Stanford University, Stanford, CA.

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

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

[Rickel and Porter, 1992] Rickel, Jeff and Porter, Bruce 1992. Automated modeling for anssering prediction questions: Exploiting interaction paths. In *Proceedings of the Sixth International Workshop on Qualitative Reasoning about Physial Systems.*

[Weld, 1990] Weld, Daniel 1990. Approximation reformulation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Los Altos, CA. Morgan Kaufmann.

[Williams, 1990a] Williams, Brian C. 1990a. Capturing how things work: Consructing critical abstractions of local interactioins. In *Proceedings of the Workshop on Automatic Generation of Approximations and Abstractions, pp.163-174.*

[Williams, 1990b] Williams, Brian C. 1990b. Interaction-based invention: Designing novel devices from first principles. In *Proceedings of the Eighth National Conference on Artificial Intelligence.*