ELSEVIER Artificial Intelligence 83 (1996) 193–227

Artificial Intelligence

Efficient compositional modeling for generating causal explanations

P. Pandurang Nayak[a,*], Leo Joskowicz[b]

[a] *Recom Technologies, NASA Ames Research Center, MS 269-2, Moffett Field, CA 94035, USA*

[b] *Institute of Computer Science, The Hebrew University of Jerusalem, Jerusalem, 91904 Israel*

Received September 1993; revised March 1995

Abstract

Effective problem solving requires building adequate models that embody the simplifications, abstractions, and approximations that parsimoniously describe the relevant system phenomena for the task at hand. *Compositional modeling* is a framework for constructing adequate device models by composing model fragments selected from a model fragment library. While model selection using compositional modeling has been shown to be intractable, it is tractable when all model fragment approximations are *causal approximations*.

This paper addresses the reasoning and knowledge representation issues that arise in building practical systems for constructing adequate device models that provide parsimonious causal explanations of how a device functions. We make four important contributions. First, we present a representation of class level descriptions of model fragments and their relationships. The representation yields a practical model fragment library organization that facilitates knowledge base construction and supports focused generation of device models. Second, we show how the structural, behavioral, and functional contexts of the device define model adequacy and provide the task focus and additional constraints to guide the search for adequate models. Third, we describe a novel model selection algorithm that incorporates device behavior with *order of magnitude* reasoning and focuses model selection with component interaction heuristics. Fourth, we present the results of our implementation that produces adequate models and causal explanations of a variety of electromechanical devices drawn from a library of 20 components and 150 model fragments.

1. Introduction

Effective problem solving about complex physical systems requires building models that are both adequate for the task and computationally efficient. Adequate models em-

* Corresponding author. E-mail: nayak@ptolemy.arc.nasa.gov.

body the simplifications, abstractions, and approximations that parsimoniously describe the relevant system phenomena for the task at hand. Overly detailed models are computationally expensive, contain irrelevant information, obscure qualitative differences, and often have unstable solutions or are altogether unsolvable. Overly simple models miss or distort key phenomena, ignore relevant interactions, and can lead to erroneous conclusions.

In current practice, most system models are either hand-crafted or are automatically derived assuming the relevant system-wide phenomena have been identified. Hand-crafting models is difficult, error-prone, and time-consuming, requiring a good understanding of what the system does, and how it does it. Recent model-based reasoning systems derive device models from a description of their structure based on so called system-wide or class-wide assumptions. These assumptions determine the simplifications, phenomena types, and interaction modes applicable to all the components of the device. For example, in modeling digital circuits, component models ignore all but the information processing capabilities. System-wide assumptions are useful but limit the modeling scope and fail in devices that exhibit many types of phenomena. The goal of automating the model construction process is to overcome these limitations and provide future programs with an effective modeling tool.

Recently, Falkenhainer and Forbus [10] proposed *compositional modeling*, a framework for constructing adequate device models by composing *model fragments* selected from a model fragment library. Model fragments are partial descriptions of components and physical phenomena embodying different assumptions, simplifications, abstractions, and approximations. Model construction is a search process, where the goal is to select an adequate model from the space of possible models defined by the library's model fragments. Falkenhainer and Forbus demonstrate compositional modeling in a tutorial setting. They develop a model composition algorithm that constructs a device model which satisfies the modeling constraints and answers the user query.

While compositional modeling is a general framework that can be used in a variety of tasks, it is also inherently intractable. Nayak [25] shows that model selection using compositional modeling becomes tractable when all model fragment approximations are *causal approximations*. Causal approximations, which are common in modeling the physical world, are based on the idea that more approximate descriptions usually explain less about a phenomenon than more accurate descriptions. When all approximations are causal approximations, the causal relations entailed by a model decrease monotonically as models become simpler. Thus, not all combinations of model fragments need to be considered when constructing an adequate model.

This paper focuses on the reasoning and knowledge representation issues that arise in building an efficient system for constructing adequate device models. Effective knowledge representation and constrained reasoning are essential to demonstrate the practical utility of compositional modeling and causal approximations. Specifically, we address the problem of constructing device models that provide parsimonious causal explanations of the functioning of a device. Causal explanations play a central role in communicating with human users and in automating a vast array of reasoning tasks, including tutoring [4,13], diagnosis [6,26], design [35], and analysis [12]. Models that support causal explanations are necessary to generate causal explanations and are useful to answer queries, run simulations, and make predictions. They can be used as base models incrementally modified for other tasks.

In this paper we make four important contributions. First, we describe a representation of class level descriptions of model fragments and their relationships. The representation provides the means for representing relevant domain knowledge and yields a practical model fragment library organization that facilitates knowledge base construction and maintenance and supports focused generation of device models. Second, we show how the structural, behavioral, and functional contexts of the device define model adequacy and provide the task focus and additional constraints to guide the search for adequate models. To model an important aspect of the functional context, we use *expected behaviors*, an abstract, causal description of *what* a device does (but not *how* it does it). Third, we describe a novel model selection algorithm that incorporates device behavior using *order of magnitude* reasoning and focuses model selection using the *component interaction heuristic*. Our method extends previous model composition algorithms [10,25] by incorporating behavior into the model selection process, by starting with a simpler initial device model, and by only instantiating model fragments that are used in the model construction process. Fourth, we present experimental results of our implementation that produces adequate models and causal explanations of a variety of electromechanical devices drawn from a library of 20 component and 150 model fragments. The program constructs models for devices with 10–54 components in 0.5–8 minutes on a workstation.

The rest of this paper is organized as follows: Section 2 illustrates compositional modeling with an example and motivates our approach. Section 3 briefly reviews compositional modeling, causal approximations, and model parsimony. Section 4 describes the model fragment classes and their organization into a library. Section 5 defines task adequacy with expected behaviors, defines the structural and behavioral contexts, and identifies the types of constraints associated with them. Section 6 describes order of magnitude reasoning and its role in model selection, introduces the component interaction heuristic, and describes the model selection algorithm. Section 7 presents experimental results from the implementation. Section 8 discusses related work and Section 9 concludes with extensions and future work.

2. Example: modeling a temperature gauge

Fig. 1 shows the schematic of a temperature gauge. It consists of a battery, a wire, a bimetallic strip, a pointer, and a thermistor. A thermistor is a semiconductor device; a small increase in its temperature causes a large decrease in its resistance. A bimetallic strip has two strips made of different metals welded together. Temperature changes cause the two strips to expand by different amounts, causing the bimetallic strip to bend. The function of the temperature gauge is to measure the temperature of a liquid in a container by posting its value on a scale. An important aspect of its function is captured by the following expected behavior: the temperature of the thermistor determines the angular position of the pointer.

The temperature gauge achieves its expected behavior as follows: the thermistor senses the water temperature. The thermistor temperature determines the thermistor resistance,

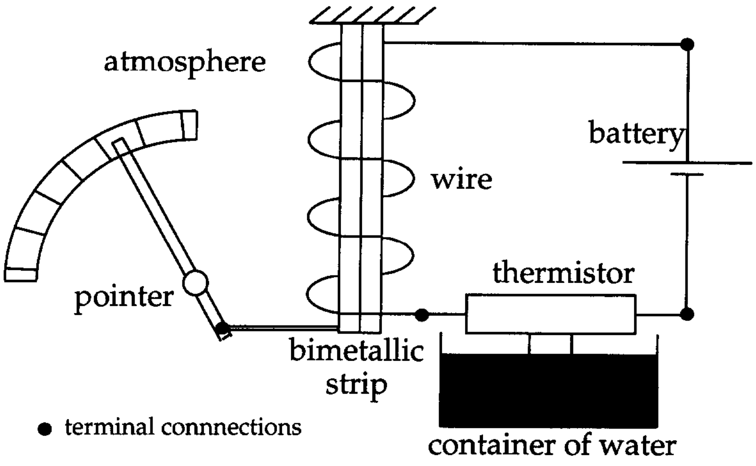

Fig. 1. A temperature gauge.

which determines the circuit current. This current generates heat in the wire, which determines the temperature of the bimetallic strip. This determines the deflection of the free end of the bimetallic strip, and hence the position of the pointer along the scale.

The main difficulty in modeling this device is accounting for the variety of physical phenomena involved (electrical, thermal, magnetic, kinematic), assessing their relative importance, and selecting among the many different ways in which each component can be modeled and can interact with its neighbors. In compositional modeling, components and their interactions are represented using model fragments that are organized in a model fragment library. For example, Fig. 2 shows a hierarchy with some of the possible model fragments that can be used to model a wire. The wire can be modeled as an electrical conductor, an electromagnet, or an inductor. When it is modeled as an electrical conductor, it can also be modeled either as an ideal conductor or as a resistor. In the latter case, its resistance can be modeled as being constant, or as being dependent upon its temperature. When the wire is modeled as a resistor, it can also be modeled as a thermal resistor, which models the heat generated in the wire due to current flow. Finally, as with most other components, the wire can be modeled using various thermal, mass, and motion models. Similarly, the wire and the bimetallic strip can interact with each other in a number of ways: thermally (heat generated in the wire can cause the bimetallic strip to heat up); magnetically (a magnetic field generated in the wire can cause the bimetallic strip to be magnetized); and kinematically (translation or rotation of the bimetallic strip causes the same motion in the wire).

An adequate model for the temperature gauge models the battery as a constant voltage source, the wire as a resistor that generates heat, the pointer as a rotating object, and the bimetallic strip, thermistor, and atmosphere as thermal objects with appropriate models. The resulting model is shown in Fig. 3. It includes model fragments for the components of the temperature gauge (e.g., the wire `wire-1`) and for the component interactions (e.g., `atmosphere-bms` represents interactions between the atmosphere and the bimetallic strip).

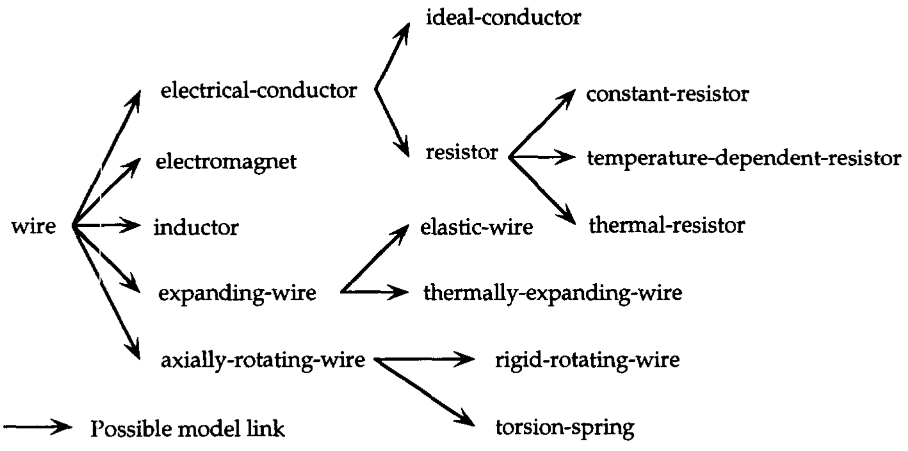

Fig. 2. Possible models of a wire. The hierarchy shows the different model fragments that can be used, some inclusively, some mutually exclusively, to model a wire.

This model satisfies the three important properties of an adequate model. First, it is able to explain the expected behavior of the temperature gauge. Fig. 4 shows the causal ordering generated from the equations of this model. It matches the explanation of the expected behavior presented earlier. Second, it includes all significant phenomena and leaves out insignificant phenomena. For example, the internal resistance of the battery is insignificant, and is not included in the model. Third, it excludes all irrelevant phenomena, so that it is as simple as possible. For example, the electromagnetic field generated by the wire is irrelevant, and has been excluded.

3. Background

Our starting points are compositional modeling [10] and causal approximations [25]. This section briefly summarizes their key concepts and describes the notion of model parsimony as defined in [25]. Our focus is on behavioral device models: behavioral models describe relations between the device's physical attributes, and how these attributes' values evolve over time.[1] Models of device behavior are described by sets of equations that relate sets of device parameters. We restrict ourselves to time-varying and equilibrium lumped parameter models represented using ordinary differential equations, algebraic equations, and qualitative equations [3,33].

3.1. Compositional modeling

Compositional modeling constructs device models by composing *model fragments*. A model fragment is a set of non-redundant equations that partially describes some phys-

[1] Unlike structural models, which are often directly available from CAD tools, and functional models, which are typically part of the design description, behavioral models must be constructed. Henceforth, unless otherwise noted, we will use the term "model" to refer to behavioral models.

`(Constant-temperature atmosphere-1):`	$exogenous(T_a)$,
`(Constant-temperature-model thermistor-1):`	$exogenous(T_t)$,
`(Thermal-thermistor thermistor-1):`	$V_t = i_t R_t;\quad M-(R_t, T_t)$,
`(Constant-voltage-source battery-1):`	$exogenous(V_v)$,
`(Constant-resistance wire-1):`	$exogenous(R_w)$,
`(Resistor wire-1):`	$V_w = i_w R_w$,
`(Thermal-resistor wire-1):`	$f_w = V_w i_w$,
`(Equilibrium-thermal-model wire-1):`	$f_{wb} = f_w$,
`(Thermal-bimetallic-strip bms-1):`	$x_b = k_2 T_b$,
`(Equilibrium-thermal-model bms-1):`	$f_{ba} = f_{wb}$,
`(Resistive-thermal-conductor atmosphere-bms):`	$f_{ba} = k_3(T_b - T_a)$,
`(Resistive-thermal-conductor bms-wire):`	$f_{wb} = k_4(T_w - T_b)$,
`Kirchhoff's laws:`	$V_v = V_w + V_t;\ i_v = i_t;$ $i_t = i_w;\ \theta_p = k_1 x_b$.

θ_p:	Pointer angle	x_b:	Bms deflection
R_w:	Wire resistance	R_t:	Thermistor resistance
i_t:	Thermistor current	V_t:	Thermistor voltage
i_w:	Wire current	V_w:	Wire voltage
i_v:	Battery current	V_v:	Battery voltage
T_b:	Bms temperature	T_a:	Atmosphere temperature
T_w:	Wire temperature	T_t:	Thermistor temperature
f_{ba}:	Heat flow (bms to atmosphere)	f_{wb}:	Heat flow (wire to bms)
f_w:	Heat generated in wire	k_j:	Exogenous constants

Fig. 3. Model fragments and equations describing the temperature gauge. The equation $exogenous(Q)$ indicates that the value of Q is determined exogenously. The equation $M-(Q_1, Q_2)$ is a qualitative equation indicating that Q_1 decreases when Q_2 increases [18].

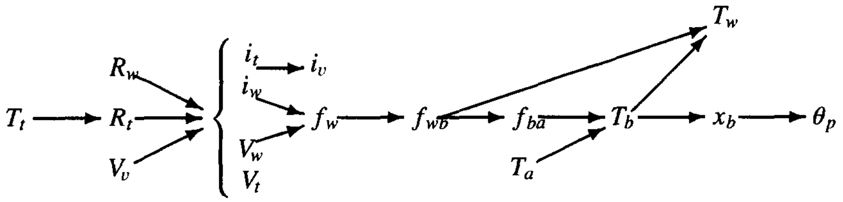

Fig. 4. The causal ordering generated from the model in Fig. 3. Directed arrows show the causal dependency relation between parameters. Bracketed parameters are determined simultaneously.

ical phenomenon at some level of granularity. Different model fragments can describe different phenomena (e.g., electrical conduction, heat generation, electromagnetism) or can be different descriptions of the same phenomenon (e.g., describing electrical conduction in a wire as a resistor, an ideal conductor, or an ideal insulator). Since model fragments are only partial descriptions of phenomena, additional model fragments are necessary to complete the description. For example, modeling a wire as a resistor requires additional model fragments describing the variation of the wire's resistance. The set of model fragments represents the space of possible device models.

The equations of a device model are constructed by composing the equations of the model fragments. For algebraic and differential equations, the composition is a union of the equations in the model fragments. Qualitative equations and expressions with special operators, e.g., direct ($I\pm$) and indirect influences ($\alpha_{Q}\pm$) [11], are composed with special rules.

Using model fragments to represent the space of device models has two advantages. First, the set of model fragments is a representation of an exponentially large set of device models: any subset of it can be composed to form a model. This representation, unlike the explicit representation of each model [1], allows broad coverage of phenomena and scales up. Second, model fragments are highly reusable, both in different models of the same device and in models of different devices. Thus, the effort of constructing a library of model fragments can be amortized over their use in different models.

3.2. Causal approximations

Compositional modeling is, in general, intractable. Nayak [24,25] formalizes compositional modeling and shows that finding adequate models using compositional modeling is NP-hard. To make it tractable, Nayak introduces a new class of model fragment approximations called *causal approximations*. Causal approximations have the property that when all approximations are causal approximations, model selection using compositional modeling does not require exhaustive search.

Causal approximations are based on the observation that more approximate descriptions often involve fewer parameters and explain less about a phenomenon than more accurate descriptions. Formally, causal approximations are defined as follows: every equation in the more approximate model fragment must have a corresponding equation in the more accurate model fragment which uses a superset of parameters. Hence, when all approximations are causal approximations, the causal relations entailed by a model decrease monotonically as model fragments in the model are replaced by their approximations. Modeling the different aspects of the physical world with causal approximations is both natural and commonplace in physics and engineering. Many examples of causal approximations, such as elastic collisions, frictionless motion, and ideal gas laws are described in [25].

3.3. Model simplicity

An adequate model must parsimoniously describe the relevant system phenomena for the task at hand. Hence, any notion of model adequacy must incorporate some notion of model simplicity. Following [25], we establish a partial order between models based on the approximation relation between model fragments. A model is usually simpler if it models fewer phenomena or does so more approximately. Formally, we say that model M_2 is *simpler* than model M_1 if and only if for each model fragment $m_2 \in M_2$ either (a) $m_2 \in M_1$; or (b) there is a model fragment $m_1 \in M_1$ such that m_2 is an approximation of m_1.

For example, a simpler model of the temperature gauge model in Fig. 3 ignores the wire's heating properties by removing the model fragment (`Thermal-resistor` wire-1). A more complex model takes into account the thermal properties of the wire's resistance by replacing the model fragment (Constant-resistance wire-1) with the more accurate model fragment (Temperature-dependent-resistor wire-1). An incomparable model results from both removing (Thermal-resistor wire-1) and replacing (Constant-resistance wire-1) by (Temperature-dependent-resistor wire-1).

Note that this definition of model simplicity does not guarantee that simpler models will be more efficient to simulate or will produce simpler causal explanations than more complex ones. However, it is a good heuristic, following the common engineering practice of disregarding irrelevant phenomena and using all applicable approximations.

4. Representation and organization of the model fragment library

Relations between model fragments are a key part of the domain knowledge that are needed for effective model construction. Identifying and capturing these relationships is essential for representing the relevant domain knowledge and for organizing the model fragment library to focus the search for adequate models. In this section, we identify several key model fragment relations, describe how to represent them, and show how to effectively organize the model fragment library according to them. This organization provides compact representations, facilitates knowledge base construction and maintenance, and supports efficient model fragment retrieval and generation of device models. Section 6 shows how to effectively use the library in the model selection search process.

The library is organized along three major lines. First, the model fragment library contains class level descriptions of model fragments. The model fragment classes are organized into a generalization hierarchy that captures the "subset-of" relation between classes. Second, the space of possible device models is represented using a possible models hierarchy that captures the additional ways of modeling instances of a class. A set of articulation rules ensure that attributes of different model fragment classes are used coherently. Third, the representation of three relations, *contradictory*, *required*, and *approximation* ensures the consistency, completeness, and parsimony of models built from model fragments.

4.1. Model fragment classes

Model fragments in the library are represented as *classes*; a component is modeled by a model fragment by making it an instance of the corresponding class. For example, a wire, wire-1, is modeled as a resistor by making it an instance of the Resistor class. The literal (Resistor wire-1), which also denotes the corresponding model fragment, indicates this choice. A component can be simultaneously modeled by more than one model fragment by making it an instance of each of the corresponding classes.

Model fragment classes inherit properties to their instances. The properties comprise the definitions of the phenomena being modeled by the model fragment class. Two prop-

```
(defmodel Resistor
  (attributes
    (resistance
      :range Resistance-parameter
      :documentation " The resistor's resistance"))
  (equations
    (= (voltage-difference ?object)
       (* (resistance ?object)
          (current (electrical-terminal-one ?object)))))
  (generalizations Electrical-conductor)
  (possible-models Constant-resistance
                   Temperature-dependent resistance
                   Thermal-resistor)
  (assumption-class electrical-conductor-class)
  (required-assumption-classes resistance-class))
  (approximations Ideal-conductor
                  Ideal-insulator)
```

Fig. 5. The Resistor model fragment class.

erties are inherited to instances: *attributes* and *equations*. Attributes are either numerical attributes, which are the parameters of the phenomena being modeled, or terminals, which specify the ports through which components can interact with each other by sharing parameters [7]. Equations are defined using *equation schemas* that are instantiated for specific instances of the model fragment class. Equation schemas are equations whose parameters are replaced by terms such as `(resistance ?object)` and are instantiated by binding the variable `?object` to the instance and replacing the terms by the parameter resulting from evaluating the term.

Fig. 5 shows the `Resistor` model fragment class. The `attributes` clause specifies the `resistance` parameter for instances of `Resistor`. The `:range` specification defines the type of this attribute. The `equations` clause specifies the relation between the different parameters. (The remaining clauses are discussed in detail below.)

4.1.1. Generalization hierarchy

Model fragment classes are organized into a standard *generalization hierarchy* representing the "subset-of" relation between classes. Knowledge represented with a class can be used both by direct instances of the class and by instances of subclasses (specializations) of the class, thereby facilitating its reuse. Also, since knowledge needs to be represented only with the most general class to which the knowledge is applicable, changes tend to be localized, thereby facilitating knowledge base maintenance.

For example, the `generalizations` clause in Fig. 5 states that the `Electrical-conductor` class is a generalization of the `Resistor` class. Hence, any component being modeled as a resistor is also modeled as an electrical conductor.

4.2. Representing the space of device models

Model fragment classes are an effective way of compactly representing the set of model fragments, which define the set of device models (Section 3.1). To generate the set of model fragments from these classes requires knowing which model fragment classes can be used to model each component. For example, we must represent the fact that a wire can be modeled as a resistor but not as a thermal thermistor, which models the dependence of a thermistor's resistance on its temperature. Furthermore, to ensure that the model description is complete, we must represent how attributes in model fragments relate and complement each other. For example, `wire-1` inherits the attributes `wire-terminal-one` and `wire-terminal-two`, representing the two ends of the wire, from `Wire`. When it is modeled as an `Electrical-conductor` it also inherits the attributes `electrical-terminal-one` and `electrical-terminal-two`, corresponding to the two ends of the electrical conductor. Since the ends of the electrical conductor are identical to the ends of the wire, they must be considered as identical.

4.2.1. Possible models hierarchy

To represent the set of model fragment classes that can be used to model each component, we organize the model fragment classes into a *possible models hierarchy*. The possible models of a model fragment class are the additional ways of modeling instances of that class. The transitive closure of the possible models of a class is the set of all possible ways of modeling instances of that class. For example, Fig. 2 shows part of the possible models hierarchy rooted at `Wire`. The `possible-models` clause in Fig. 5 specifies that instances of `Resistor` can also be modeled as instances of `Constant-resistance`, `Temperature-dependent-resistance`, and `Thermal-resistor`.

The possible models hierarchy has advantages similar to the generalization hierarchy. First, it leads to compact representations. For example, we only need to specify that instances of `Wire` can be modeled as an `Electrical-conductor`. The additional ways of modeling instances of `Wire` are directly inferred from the hierarchy. Second, it simplifies knowledge base maintenance. For example, adding a model fragment class describing the dependence of resistance on length only requires changing the possible models hierarchy below `Resistor`, while model fragment classes above it, such as `Wire` remain unchanged.

The generalization hierarchy and the possible models hierarchy often overlap, but they are not the same. For example, `Resistor` is both a specialization and a possible model of `Electrical-conductor`. However, the `Thermal-thermistor` model fragment class is a specialization of the `Thermal-object` model fragment class, but not all components being modeled as `Thermal-objects` can be modeled as `Thermal-thermistors`. The generalization hierarchy captures the "subset-of" relation between classes, while the possible models hierarchy captures additional ways to modeling instances of a class.

4.2.2. Articulation rules

To represent the relations between attributes introduced by different model fragment classes we introduce *articulation rules* (similar to articulation axioms in [15]). These rules ensure that all relations between attributes of different model fragment classes are correctly represented. For example, the relations between the wire terminals and the electrical terminals, discussed above, are enforced by the rule:

```
(implies
   (and (Wire ?object)
        (Electrical-conductor ?object)
        (wire-terminal-one ?object ?term1)
        (wire-terminal-two ?object ?term2))
   (and (electrical-terminal-one ?object ?term1)
        (electrical-terminal-two ?object ?term2)))
```

This rule captures the fact that the two ends of the wire are identical to the two ends of the electrical conductor so that components connected to the ends of `wire-1` will be able to electrically interact with it.

4.3. *Representing model consistency, completeness, and parsimony*

The possible models hierarchy represents the space of possible device models. However, not all device models in this space are consistent, complete, and parsimonious. For example, a wire can be modeled as an ideal conductor or as an ideal insulator, but not both. Thus, a consistent device model can contain at most one of these model fragments. Similarly, modeling a wire as a resistor requires a model of the resistance. Thus, a complete device model must include both model fragments. Finally, modeling the internal resistance of a battery is superfluous if the battery is modeled as a constant voltage source. Thus, a parsimonious device model that includes the latter model fragment would not include the former.

We ensure that device models are consistent, complete, and parsimonious by representing additional relations between model fragments. We represent the contradictory relation between model fragments by organizing mutually contradictory model fragments into *assumption classes*. We represent the set of model fragments that can be used to complete the (partial) description in a model fragment by introducing the assumption classes *required* by a model fragment. Finally, we show how the approximation relation, introduced earlier in our discussion on model parsimony, is represented.

4.3.1. *Assumption classes*

When model fragments describe the same phenomenon, they often make mutually contradictory assumptions. Following [1, 10, 25], we represent the contradictory relation between model fragments by grouping mutually contradictory model fragments into disjoint *assumption classes*. To avoid inconsistencies, consistent models must include at most one model fragment from each assumption class. The explicit representation of assumption classes makes it easy to generate consistent device models.

For example, the following three model fragments describing electrical conduction in a wire:

(Ideal-conductor wire-1): $V_w = 0$,
(Ideal-insulator wire-1): $i_w = 0$,
(Resistor wire-1): $V_w = i_w R_w$,

are mutually contradictory: the ideal conductor model fragment assumes that the resistance of the conductor is zero, the ideal insulator model fragment assumes that the resistance of the conductor is infinite, and the resistor model fragment assumes that the resistance of the conductor is nonzero and finite. The assumption-class clause of the model fragment classes Resistor (see Fig. 5), Ideal-insulator, and Ideal-conductor all specify electrical-conductor-class.

Note that the contradiction between model fragments in an assumption class cannot, in general, be derived from the equations of the model fragments. For example, the equations of the ideal conductor model fragment and the ideal insulator model fragment can be simultaneously satisfied when both the current through a conductor and the voltage drop across it are zero. However, these model fragments are mutually exclusive because their underlying assumptions are contradictory. Assumption classes capture this domain-dependent fact.

4.3.2. Required assumption classes

Since model fragments are partial descriptions of phenomena, additional model fragments are sometimes required to complete their description. We represent the set of model fragments that can be used to complete a description by introducing the assumption classes *required* by a model fragment; the description of the model fragment is completed by including a model fragment from each required assumption class. This representation makes it easy to generate device models that include complete descriptions of all modeled phenomena. We will use this fact in our model selection algorithm.

For example, the required-assumption-classes clause in Fig. 5 shows that a component being modeled as a Resistor must also be modeled using a model fragment class that specifies Resistor-class as its assumption-class, viz. Constant-resistance or Temperature-dependent-resistance.

4.3.3. Approximation relation

We define model parsimony based on the approximation relation between model fragments (Section 3). When one model fragment is an approximation of another, they clearly make mutually contradictory assumptions, and hence belong in the same assumption class. Hence, we represent the approximation relation between model fragments by organizing the model fragments within an assumption class into an approximation hierarchy. For example, the approximations clause in Fig. 5 specifies that Ideal-conductor and Ideal-insulator are more approximate descriptions of electrical conduction than Resistor.

The representation of the approximation relation as a hierarchy makes it easy to minimally simplify a model by replacing a model fragment by one of its immediate approximations. This is useful since our model selection algorithm (see Section 6) is based on the ability to minimally simplify a model.

5. Adequate models

In addition to being consistent, complete, and parsimonious, device models must also be adequate for the task at hand. For example, to analyze the performance of the temperature gauge in Fig. 1 during the final stages of detailed design, it is necessary to include complex nonlinear differential equations describing the dependence of the thermistor's resistance on its temperature. However, a simple qualitative current/no current model is sufficient for determining why the pointer does not move when the water temperature rises. Model adequacy is critically dependent on the context in which the device operates. In this section, we introduce the functional, structural, and behavioral contexts that define model adequacy.

The functional context is an abstract description of the aspect of the device's functioning that is of interest. The functional context determines the set of relevant phenomena, and the level of granularity that must be included in an adequate model. The structural context is defined by the device's components, their physical and structural properties, and structural relations between them describing how they are put together to form the device. Together with the model fragment library, it defines the space of possible models and provides constraints that guarantee structurally coherent models. The behavioral context is defined by the values, and the variations over time of the values, of the physical parameters used to model the device. It provides constraints that ensure that all significant phenomena are modeled with applicable approximations.

5.1. Functional context

The functional context is provided by the function of the device that is of interest. The function is an abstract description of some aspect of *what* the device does. An adequate model must be able to explain *how* the device achieves this function. The most common functional descriptions are input/output descriptions of device behavior, and often correspond to the primary function of the device. Knowledge of device function is commonplace and almost always available either directly from the user, from the description of the problem to be solved, or from the context in which the device operates. For example, the primary function of the temperature gauge in Fig. 1 is to measure temperature by measuring the deflection of the pointer as the temperature of the liquid changes.

Device function can be represented at different levels of abstraction [5]. For example, the relationship between the pointer's angular position and the temperature can be represented as a causal relation, a qualitative relation, or an algebraic relation. The "right" level of abstraction should be determined by the task at hand. In this paper we focus on the task of generating parsimonious causal explanations for the functioning of the device [25]. We specify device function using causal relations between parameters; an adequate device model must provide a parsimonious explanation for these causal dependencies. We call such required causal relations the *expected behavior* of the device. For example, the expected behavior of the temperature gauge, representing its primary function, is:

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

which states that an adequate device model must explain how the temperature of the thermistor causally determines the angular position of the pointer.

We test if a model satisfies the expected behavior by generating the *causal ordering* of the parameters of the model, using the equations of the model [7,11,17,30,34]. The causal ordering can be generated efficiently using the causal ordering algorithm discussed in [28]. The causal ordering identifies causal dependencies between the parameters of the model, and hence can be directly used to check if the model explains the expected behavior. For example, recall that Fig. 4 shows the causal ordering generated from the equations of the model in Fig. 3. This causal ordering provides an explanation for the above expected behavior.

5.2. *Structural context*

The structural context is defined by the components of the device, their physical and structural properties, and their structural relations. The user models the structure of the device by selecting model fragments from the library and by specifying their properties and the structural relations between them. The structural context provides the basis for model construction by defining the space of possible component models and their interactions.

The particular choice of components in the library, their properties, and the possible structural relations reflects the domain of interest and defines the most detailed level of granularity that needs to be considered. For example, in the electro-mechanical domain, the components of interest include wires, batteries, magnets, and springs. The physical and structural properties include shape, dimension, mass, and material composition. Structural relations, which describe how components are put together, include `connected-to`, indicating that two component terminals are connected to each other [7], `coiled-around`, indicating that a wire is coiled around a component, `meshed` indicating that a pair of gears mesh with each other, and `immersed-in`, indicating that a component is immersed in a fluid. These structural relations determine how components may interact, e.g., the `connected-to` relation supports electrical, thermal, and kinematic interactions between components. The structural context of a device can change during the operation of the device, as new components are created and old ones are destroyed (e.g., boiling water becoming steam), as the physical properties of components change (e.g., demagnetizing the magnetic strip on a credit card), or as structural relations between components change (e.g., the intermittent contact between the hammer and the dome of an electric bell).

To ensure structurally coherent device models and to model structural context changes, we associate structural constraints with model fragment classes. These constraints state that a component can be modeled by a model fragment class only if it satisfies all the structural constraints associated with that class. We distinguish between two types of constraints: *structural preconditions* and *structural coherence constraints.*

Structural preconditions are necessary constraints on the structural context that must be satisfied if a component is to be modeled by the model fragment class. For example the precondition:

```
(and (composition ?object ?material)
     (metal ?material))
```

in the Electrical-conductor model fragment class indicates that a component must be metallic for it to be modeled as an electrical conductor. Structural preconditions are similar to process preconditions in QP theory [11], except that structural preconditions are necessary conditions, while process preconditions are sufficient conditions.

Structural coherence constraints restrict the model fragment classes that can be used to model a set of structurally related components. For example, the constraint:

```
(implies
   (and (Wire ?object)
        (coiled-around ?object ?core)
        (magnetic-material ?core))
   (Magnet-class ?core))
```

associated with the Electromagnet model fragment class specifies that a wire coiled around a core made of magnetic material can be modeled as an electromagnet only if the core is also modeled as a magnet (i.e., by a model fragment class in the Magnet-class assumption class). This is because the core amplifies the wire's magnetic field by three or four orders of magnitude, converting the core into a powerful magnet. Without this amplification, the magnetic effect is considered negligible.

5.3. Behavioral context

The behavioral context of a device is defined by the values of the physical parameters used to model the device; it changes over time with the parameter values. For example, the behavioral context of the temperature gauge in Fig. 1 includes the values for the current flowing in the circuit, the magnetic field generated by the wire, and the angular position of the pointer.

Ideally, we would like the behavioral context to refer to the actual behavior of the device, e.g., to the values of the parameters obtained from actual measurements on a physical prototype. Since the actual device behavior is usually unavailable, the behavior must be computed from the device model equations. Because different device models can predict different behaviors and introduce different errors, it is important to use the device model that introduces the least error. Hence, we use the behavior predicted by the most accurate (complex) device model as the behavioral context.

To ensure that models have acceptable accuracy, we introduce two types of domain-dependent behavioral constraints: *behavioral preconditions*, indicating which approximations are acceptable, and *behavioral coherence constraints*, indicating which phenomena are significant. Behavioral preconditions are constraints associated with model fragment classes that must be satisfied if a component is to be modeled by that model fragment class. They specify necessary conditions under which an approximate model fragment class can model a component. For example, the behavioral precondition:

```
(< (voltage-difference ?object)
   (voltage-difference-threshold ?object))
```

associated with the `Ideal-conductor` model fragment class indicates that a component can be modeled as an ideal conduct, rather than more accurately as a resistor, only if the voltage drop across it is less than some threshold. Behavioral preconditions are similar to process quantity conditions in QP theory [11]. However, behavioral preconditions are modeling constraints used to decide which model fragment classes in an assumption class can model a component, while quantity conditions control the activity of a process, and are about the physics of the situation.

Behavioral coherence constraints, like structural coherence constraints, constrain the model fragment classes that can model a set of related components. They ensure that device models include all significant phenomena. For example, the constraint:

```
(implies
   (>= (* (voltage-difference ?object)
          (current (electrical-terminal-one ?object)))
       (electrical-power-threshold ?object))
   (Thermal-resistor-class ?object))
```

in the `Resistor` model fragment class specifies that when the power dissipation of a component modeled as a resistor exceeds a prespecified threshold, this power dissipation is deemed significant and must be explicitly modeled by adding a model fragment class from the `Thermal-resistor-class` assumption class to the wire model.

Parameter thresholds, such as the electrical power dissipation threshold in the previous example, play a central role in determining the significance of phenomena and the applicability of approximations. Thresholds can either be preset or computed dynamically. Preset thresholds can be derived from physics, such as the Reynolds number in fluid dynamics indicating when laminar flow becomes turbulent, or can be set by an engineer depending on the application. For example, voltage differences of up to 10 volts can be considered insignificant in a power distribution system, while voltage differences of only up to .01 volts can be considered insignificant in an electronic circuit. Thresholds can also be set dynamically, based on knowledge of acceptable error tolerances on parameters. The error tolerances can be propagated, via a set of rules or through the model equations, to set other thresholds [22,29].

6. Model selection algorithm

We now present the model selection algorithm. Given a device description, its expected behavior, and a library of model fragments, we construct an adequate model by composing a set of model fragments describing the device's components and their interactions. Following [25], we identify an adequate model by first constructing an initial model that satisfies all the model adequacy criteria with the exception of model parsimony and then progressively simplifying this model until none of its immediate simplifications explains the expected behavior.

We make two contributions to the basic model selection algorithm. First, we show how to construct an initial model that is simpler than the most accurate model by using the expected behavior, the structural and behavioral constraints, and a special set of heuristic constraints called the *component interaction heuristic constraints*. This improves the basic algorithm in [25], which starts with the most accurate model as the initial model. Second, we explicitly incorporate behavior generation into our model selection algorithm using a logarithm-based *order of magnitude reasoning* technique. Behavior generation is a prerequisite for evaluating behavioral constraints.

To ensure that the initial model can be efficiently simplified, we require all model fragment approximations to be causal approximations. This guarantees that the causal relations entailed by a model decrease monotonically as it is simplified. Hence, simplifying a model that explains the expected behavior until none of its immediate simplifications does yields an adequate model. In addition, we restrict the expressivity of the structural and behavioral coherence constraints by requiring them to be rules whose antecedents are conjunctions of positive literals (representing structural properties of components, ordinal relations between quantities, or components being modeled as instances of model fragment classes) and whose consequents are positive literals representing components being modeled by an assumption class. All the coherence constraints in Section 5 have this restricted form.

This section describes the model selection algorithm. We begin by describing order of magnitude reasoning for behavior generation and the component interaction heuristic constraints. We then describe the steps of the algorithm in detail and illustrate them on the temperature gauge.

6.1. Order of magnitude reasoning

Generating device behavior is necessary to evaluate the behavioral constraints and to determine the significance of different physical phenomena. Since the device model equations can seldom be solved in closed form, we must use either numerical or qualitative methods. Numerical methods require exact values for exogenous parameters, which are not always available, and can be unstable, inefficient, or converge to the wrong solution. Qualitative methods, which consider only parameter signs [3,33], lack the discriminatory power to estimate the significance of phenomena and can lead to ambiguity. Order of magnitude reasoning, which focuses on the relative signs and magnitudes of the parameters, strikes a balance between these two extremes.

We use the order of magnitude reasoning technique embodied in a program called NAPIER [23]. NAPIER defines the order of magnitude of a quantity on a logarithmic scale and uses a set of rules to propagate orders of magnitudes through equations. It handles nonlinear simultaneous equations and uses approximation techniques to make the computation efficient. We use the computed orders of magnitude to evaluate behavioral preconditions and coherence constraints. These determine which model fragments should or should not be part of the device model.

Following is a brief description on NAPIER's operation (see [23] for a detailed account). NAPIER defines the order of magnitude of a parameter q (denoted $om(q)$) as an interval on a logarithmic scale:

$$om(q) = \lfloor \log_{10} |q| \rfloor. \qquad (1)$$

It applies rules to propagate orders of magnitude through equations. The orders of magnitude are propagated by first converting each equation into a disjunction of sets of linear inequalities and solving the resulting disjunctive linear program. For example, the equation

$$q_3 = q_1 * q_2$$

yields two linear inequalities (in this case, a single disjunct):

$$\{om(q_1) + om(q_2) \leqslant om(q_3), om(q_3) \leqslant om(q_1) + om(q_2) + 1\}.$$

NAPIER then solves the disjunctive logic programs using linear programming and backtracking. Since solving the disjunctive logic program is, in general, NP-hard, NAPIER uses a heuristic method based on causal ordering. The heuristic method is fast and does not appear to lose accuracy in practice. For example, NAPIER solves a set of 163 equations in only 21 seconds on a workstation, with no loss of accuracy.

To illustrate how orders of magnitude are used in model selection, consider modeling a wire through which current is flowing. Suppose that NAPIER has predicted that the order of magnitude of the current through the wire is -1 (several deciamperes), and that of the voltage across the wire is 0 (several volts). To determine if the heat generated in the wire is significant and must be modeled, recall the behavioral coherence constraint in Section 5.3:

```
(implies
   (>= (* (voltage-difference ?object)
          (current (electrical-terminal-one ?object)))
       (electrical-power-threshold ?object))
   (Thermal-resistor-class ?object))
```

where the order of magnitude of the electrical power threshold is -1 (several deciwatts). Following the previous order of magnitude rule, the order of magnitude of the product of the current and the voltage is between -1 and 0, which is greater than or equal to the electrical power threshold. This indicates that the heat generated by the wire is significant and hence the wire should be modeled as a thermal resistor.

6.2. Component interaction heuristic

To focus the search for initial adequate models we introduce the *component interaction heuristic*. The idea is to have a set of constraints on component interactions that focus the initial model selection process. However, these constraints do not define model adequacy, so the final adequate model need not satisfy them: the interactions required by these constraints might eventually be discarded because they may be too weak to be worth modeling or may not be relevant in explaining the expected behavior.

Components can only interact with each other if the components are related by the structural relations that support the interaction and the component models are compatible with the interaction. For example, two wires can interact electrically only if they are connected to each other and if they are both modeled as electrical conductors. The heuristic requires that if a set of components are related by one or more structural relations that support an interaction, and if one of the component models is compatible with this interaction, then the remaining component models must be augmented to be compatible with this interaction. This allows the components in the set to interact with each other via that interaction. Note that if none of the component models is compatible with the interaction, then no augmentations are necessary.

We implement the component interaction heuristic with a set of constraints. Each constraint specializes the heuristic for a particular interaction and structural relations. For example, the constraint:

```
(implies
   (and (terminals ?object ?term1)
        (voltage-terminal ?term1)
        (connected-to ?term1 ?term2)
        (terminal-of ?term2 ?comp2))
   (electrical-component ?comp2))
```

in the `electrical-component` model fragment class says that if a component is being modeled as an `electrical-component`, and one of the component's voltage terminals is connected to a terminal of another component, then the other component must also be modeled as an `electrical-component`. This allows the two components to interact by sharing voltages at the connected terminals. We require the initial model to satisfy all the component interaction constraints.

6.3. Model composition algorithm

Fig. 6 shows the algorithm for finding a device model that explains the expected behavior of a device. The inputs to the algorithm are:

- the structure of the device: its components, their physical and structural properties, and the structural interconnections between them;
- the expected behavior of the device;
- orders of magnitudes of thresholds and exogenous parameters;
- an optional set of model fragment classes preselected for each component, corresponding to modeling decisions made by the user; and
- the library of model fragments organized as described in Section 4.

The algorithm constructs an initial device model and then simplifies it. The initial device model is constructed by augmenting the initial device description to include all expected behavior parameters (step 1) and to satisfy the structural coherence and component interaction constraints (step 2). The device behavior is then generated on the current choice of model fragments using NAPIER (step 3). Its results are used to

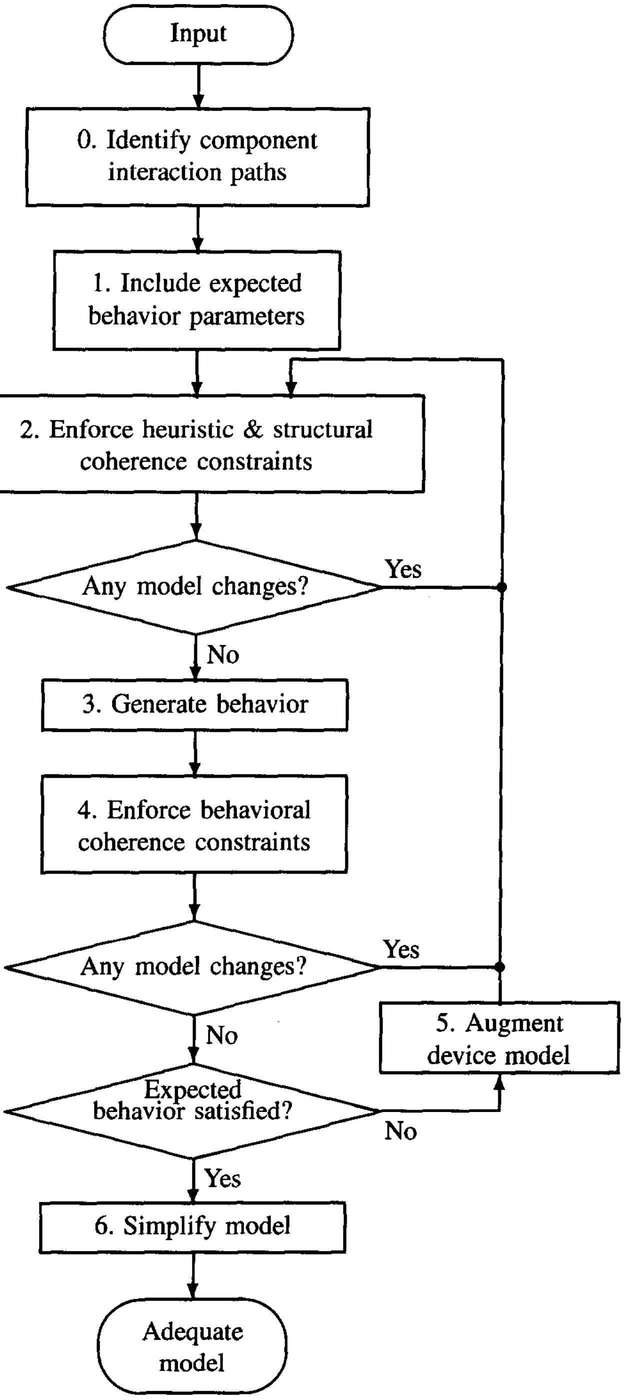
Input
0. Identify component interaction paths
1. Include expected behavior parameters
2. Enforce heuristic & structural coherence constraints
Any model changes?
Yes
No
3. Generate behavior
4. Enforce behavioral coherence constraints
Any model changes?
Yes
No
5. Augment device model
Expected behavior satisfied?
No
Yes
6. Simplify model
Adequate model

Fig. 6. Model composition algorithm.

augment the device model using the behavioral coherence constraints (step 4). If the resulting model does not explain the expected behavior, it is augmented with alternate model fragments (step 5). These steps are repeated until an initial model is found. The initial model is then simplified by either replacing model fragments with one of their approximations or by dropping them altogether (step 6).

All steps except behavior generation (step 3) and model simplification (step 6) modify the device model by adding one or more model fragments to it. When a model fragment is added to the device model, the most accurate model fragment from every assumption class required by the model fragment is also added to the model. Thus, at the end of every step the device model is complete, i.e., the device model contains a model fragment from all required assumption classes.

We now describe each step in detail. As a preprocessing step, the algorithm starts by identifying all possible component interaction paths induced by particular structural configurations with a set of rules (step 0). For example, in the temperature gauge, the wire can interact with the bimetallic strip since the former is coiled around the latter. Potential component interaction paths are considered to be additional device components and are treated exactly like the original components by the remaining steps in the algorithm, e.g., the component representing the interaction between the wire and the bimetallic strip can be modeled as a thermal or an electromagnetic interaction. Path identification is analogous to process instantiation in [11].

In step 1, the algorithm checks if the input device model contains all expected behavior parameters. If a component parameter is missing, the algorithm searches the possible models hierarchy rooted at that component for a model fragment class that provides the required parameter and whose structural preconditions are all satisfied. The resulting model fragment class is added to the component's model. When several model fragment classes satisfy the above condition, we prefer (a) more general model fragment classes over more specific ones to ensure that minimal modeling commitments are made; and (b) only most accurate model fragment classes, i.e., only model fragment classes that are not approximations of any other model fragment class (this is necessary because we cannot determine the applicability of approximations by evaluating behavioral preconditions since the behavior has not been generated).

In step 2, the algorithm checks the structural coherence constraints and the component interaction constraints of each component model. An unsatisfied constraint indicates that the component model does not include a required model fragment. The algorithm then searches the possible models of that component for a model fragment class that, when added to the component model, would satisfy the violated constraint. As before, the algorithm only considers most accurate model fragment classes whose structural preconditions are satisfied. The component model is augmented with the resulting model fragment class. This step is repeated until all structural and component interaction constraints are satisfied.

In step 3, the algorithm uses the current device model to generate the device behavior using NAPIER. NAPIER computes the orders of magnitude of the model's parameters from the equations of the model and the exogenous parameter values provided as input by the user. Note that these computed values of the model's parameters are the same as the values computed using the most accurate model. The reasons are two-fold: (a) since the current model only consists of most accurate model fragments, it is a subset of the most accurate model; and (b) the current model is complete, and the predictions of a complete model are unchanged with the addition of extra equations. Hence, the behavior predicted at this step is the part of the behavioral context relevant to the current model.

In step 4, the algorithm uses the behavior generated above to enforce all the behavioral coherence constraints. This step enforces constraints exactly like step 2. Ordinal relations are satisfied by the generated behavior if the relations can be possibly true with respect to the behavior, e.g., (>= p q) is satisfied if, given order of magnitude intervals for p and q, it is possible (not necessary) that p is greater than or equal to q. If there are any changes to the device model, the algorithm loops back to the second step, to ensure that all the structural coherence constraints and the component interaction constraints are satisfied. The algorithm loops through steps 2, 3, and 4, until all structural and behavioral coherence constraints, and the component interaction constraints are satisfied.

Once all these constraints are satisfied, the algorithm checks to see if the resulting model can explain the expected behavior. This is done by generating the causal ordering of the model parameters using the efficient causal ordering algorithm presented in [28]. Using simple graph traversal, the algorithm then checks whether the causal ordering can explain the causal relations required by the expected behavior. If the expected behavior is explained by the causal ordering, the initial model has been found and can then be simplified (step 6). Otherwise, it has to be augmented.

In step 5, the algorithm augments the device model with alternate model fragment classes identified in earlier steps. These are either model fragment classes that are more specific than those chosen earlier, or other most general model fragment classes that could have been, but were not, selected. The algorithm attempts again to enforce and satisfy all constraints (by looping back to step 2), until a model that explains the expected behavior is found. If all possible ways of augmenting the device model are exhausted, the algorithm reports that the expected behavior cannot be explained. This is justified because the component interaction heuristic guarantees that the component models used in the final device model cannot interact with any other components, and hence no augmentation of the device model can lead to a model that explains the expected behavior.

The last step is model simplification. The initial model produced by the previous steps can be more complex than necessary for one of three reasons: (a) for each required assumption class, we added in the most accurate model fragment, even though a more approximate model fragment might do; (b) model fragments added to satisfy the component interaction constraints are not strictly necessary; and (c) unnecessary model fragments may have been added when augmenting the model (step 5).

We simplify the initial device model by applying one of the following two simplification operators (see [24,25] for formal details and proofs of correctness): (a) replace a model fragment by one of its immediate approximations; and (b) remove a model fragment. During the simplification process, only approximations with satisfied structural and behavioral preconditions are considered. Behavioral preconditions are evaluated with respect to the most recent behavior generation in step 3. We first simplify the model by repeatedly applying the first simplification operator, while ensuring that the expected behavior can still be explained. We assume that applying this operator does not lead to any coherence constraints being violated, i.e., there is no coherence constraint whose antecedent is satisfied by a model simpler than the initial model and whose consequent requires an assumption class not included in the initial model.[2] We then apply the second simplification operator while ensuring that the expected behavior and all structural and behavioral coherence constraints continue to be satisfied. When all immediate simplifications of a model are unable to explain the expected behavior, the algorithm terminates, and returns that model as an adequate model. It also generates a causal explanation for the expected behavior using causal ordering as described earlier.

The application of different sequences of simplification operators can result in different models. All such models can be found by the straightforward addition of backtracking into the simplification process. One can then use additional preference criteria to choose the most preferred simplest model. However, note that the different simplest models differ only in features deemed to be insignificant by the behavioral constraints and thresholds. Hence, our algorithm returns the first adequate model it finds.

In summary, the model selection algorithm finds an adequate model, i.e., a parsimonious model that explains the expected behavior and satisfies all the structural and behavioral constraints. The use of causal approximations and the restricted expressivity of the constraints ensure the correctness and efficiency of the algorithm.

6.4. Complexity of the model selection algorithm

The complexity of the model selection algorithm is determined by two factors: (a) the number of device models constructed by the algorithm; and (b) the processing of each such device model.

The number of device models constructed by the above algorithm is linear in the number of model fragments in the library. In particular, steps 1-5 only modify the model by adding zero or more model fragments. Since only consistent models are constructed, these steps can add at most one model fragment from each assumption class. Hence, the number of models constructed by this part of the algorithm is bounded by the number of assumption classes. Since the assumption classes are disjoint, it follows that the number of assumption classes, and hence the number of models constructed in steps 1-5, is bounded by the number of model fragments. Step 6 simplifies the initial model by either dropping a model fragment or by replacing a model fragment by one of its approximations. In either case, once a model fragment has been removed from the model, it is never reconsidered. Hence, the number of models constructed by step 6 is bounded by the total number of model fragments. Hence, the number of device models constructed by the above algorithm is linear in the number of model fragments.

[2] This assumption holds in our knowledge base for any initial model. One can ensure that it holds by adding heuristic coherence constraints that need only be satisfied by the initial model, but not by the final adequate model. These constraints can be derived from the structural and behavioral coherence constraints by replacing every model fragment in the antecedent by the most accurate model fragment in its assumption class.

Table 1
Components and their initial models

Component	Component classes	After step 1	After step 2
`thermistor-1`	`Thermistor`	`Thermal-object` `Dynamic-thermal-model`	`Thermal-object` `Dynamic-thermal-model`
`pointer-1`	`Pointer`	`Rotating-object`	`Rotating-object` `Thermal-object` `Dynamic-thermal-model`
`bms-1`	`Bimetallic-strip`		`Thermal-bimetallic-strip` `Dynamic-thermal-model`
`wire-1`	`Wire`		`Thermal-object` `Dynamic-thermal-model`
`battery-1`	`Battery`		`Thermal-object` `Dynamic-thermal-model`
`atmosphere-1`	`Atmosphere`		`Thermal-object` `Dynamic-thermal-model`
`bms-wire`	`Coil-structure`		`Resistive-thermal-conductor`
`atmosphere-pointer`	`Immersion-structure`		`Resistive-thermal-conductor`
`atmosphere-bms`	`Immersion-structure`		`Resistive-thermal-conductor`
`atmosphere-battery`	`Immersion-structure`		`Resistive-thermal-conductor`

Device models can be processed in three ways: (a) enforcing coherence and heuristic constraints; (b) checking the expected behavior; and (c) generating behavior. Enforcing the coherence and heuristic constraints is efficient, only involving a depth first search of the possible models hierarchy. Checking the expected behavior is also efficient since it is done using the polynomial-time causal ordering algorithm in [28]. Generating the order of magnitude behavior is potentially exponential but fast in practice, as discussed in Section 6.1.

6.5. Example: modeling the temperature gauge

We now illustrate the model selection algorithm on the temperature gauge in Fig. 1. Part of the input, describing the components and their component classes, is shown in the first two columns of Table 1. The last four rows show the components added by the algorithm to model component interaction paths in step zero. For example, `bms-wire` is a possible interaction path between `bms-1` and `wire-1`, corresponding to `wire-1` being `coiled-around` `bms-1`. In the first step, the expected behavior of the temperature gauge:

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

requires a `temperature` parameter for the thermistor and an `angular-position` parameter for the pointer. The former can be achieved by modeling the thermistor either as a `Thermal-object` or as a `Thermal-thermistor`. The algorithm uses `Thermal-object` since it is more general. Similarly, the pointer is modeled as a

Rotating-object. Finally, Dynamic-thermal-model is added to the thermistor model, since it is the most accurate model fragment class of the assumption class required by Thermal-object.[3] The resulting model is shown in the third column of Table 1.

In the second step, because the pointer is a Rotating-object that is connected to the free end of the bimetallic strip, a component interaction constraint requires a kinematic interaction between pointer-1 and bms-1. This constraint can be satisfied by modeling the bimetallic strip as a Thermal-bimetallic-strip, which models the deflection of the free end of the bimetallic strip as a function of its temperature. As a consequence of this modeling decision, another component interaction constraint requires a thermal interaction between bms-1 and both wire-1 (via bms-wire) and atmosphere-1 (via atmosphere-bms), so that wire-1 and atmosphere-1 are modeled as Thermal-objects and bms-wire and atmosphere-bms are modeled as Resistive-thermal-conductors. Modeling atmosphere-1 as a Thermal-object requires that all objects immersed in it must also have thermal models. The resulting device model is shown in the fourth column of Table 1.

In the third step, the algorithm uses NAPIER to generate the behavior, and then proceeds to check the behavioral coherence constraints in the fourth step. None of these constraints are currently violated, so we now have a device model that satisfies the structural, behavioral, and component interaction constraints. However, the expected behavior is not satisfied, so the algorithm proceeds to the fifth step.

Recall that an alternate way of providing the temperature parameter to thermistor-1 was to model it as a Thermal-thermistor. Hence, the fifth step augments the thermistor model with this model fragment class, and returns to the second step. Since Thermal-thermistor is an electrical model, a component interaction constraint requires an electrical interaction between the thermistor and the wire and battery. This is achieved by modeling the wire as a Resistor, and the battery as a Voltage-source-with-resistance. The resulting model is used to generate the behavior, which includes calculating the wire's voltage and current. The behavioral coherence constraint:

```
(implies
   (>= (* (voltage-difference ?object)
          (current (electrical-terminal-one ?object)))
       (electrical-power-threshold ?object))
   (Thermal-resistor ?object))
```

requires that the wire must be modeled as a Thermal-resistor, since the product of the wire's voltage and current exceeds its electrical-power-threshold. With this augmentation all the structural, behavioral, and heuristic constraints are satisfied, and the expected behavior explained. The resulting initial model is shown in the second column of Table 2.

The initial model is finally simplified by approximating and dropping model fragments. For example, in the initial model the battery is a Voltage-source-with-resistance. However, the internal resistance of the battery is very small, so that the approxima-

[3] We will not mention these required assumption class augmentations in the rest of the example.

Table 2
The initial and adequate model

Component	Initial model	Adequate model
thermistor-1	Thermal-object	Thermal-object
	Dynamic-thermal-model	Constant-temperature-model
	Thermal-thermistor	Thermal-thermistor
pointer-1	Rotating-object	Rotating-object
	Thermal-object	
	Dynamic-thermal-model	
bms-1	Thermal-bimetallic-strip	Thermal-bimetallic-strip
	Dynamic-thermal-model	Equilibrium-thermal-model
wire-1	Thermal-object	Thermal-object
	Dynamic-thermal-model	Equilibrium-thermal-model
	Electrical-conductor	Electrical-conductor
	Resistor	Resistor
	Temperature-dependent resistance	Constant-resistance
	Thermal-resistor	Thermal-resistor
battery-1	Thermal-object	
	Dynamic-thermal-model	
	Voltage-source	Voltage-source
	Voltage-source-with-resistance	Constant-voltage-source
atmosphere-1	Thermal-object	Thermal-object
	Dynamic-thermal-model	Constant-temperature-model
bms-wire	Resistive-thermal-conductor	Resistive-thermal-conductor
atmosphere-pointer	Resistive-thermal-conductor	
atmosphere-bms	Resistive-thermal-conductor	Resistive-thermal-conductor
atmosphere-battery	Resistive-thermal-conductor	

tion `Constant-voltage-source` is applicable. Similarly, `wire-1`'s resistance can be assumed to be constant, rather than temperature dependent. The simplification process also determines that the thermal properties of `pointer-1` and `battery-1` are irrelevant, and hence it drops the corresponding model fragment classes from the model. The adequate model, resulting from this simplification process, is shown in the third column of Table 2.[4]

7. Implementation and results

We have implemented the model composition algorithm in Common Lisp and tested it on a variety of electromechanical devices. This section presents the results of the implementation.

We constructed a library of 20 components, such as wires, bimetallic strips, springs, and permanent magnets. The library consists of about 150 model fragment classes including descriptions of electricity, magnetism, heat, elasticity, and the kinematics and

[4] This column includes model fragments that only provide parameters, but do not directly introduce equations into the model (e.g., `(Rotating-object pointer-1)`). Hence, the model fragments in this column are a superset of the model fragments listed in Fig. 3.

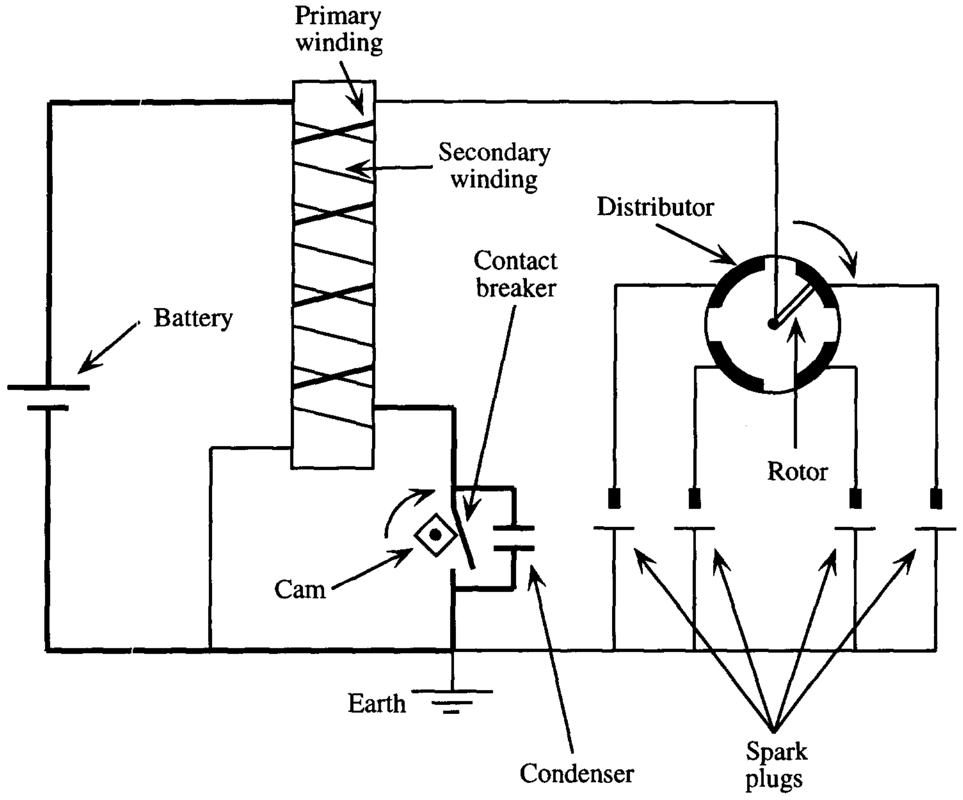

Fig. 7. Car distributor system.

dynamics of fixed rotations and translations. Each component class has an average of 30 model fragment classes describing different aspects of its behavior.

We chose ten electromechanical devices from several encyclopedias [2,20,31]. We carefully selected these devices to have similar components that needed different models. The devices range in complexity from 10 to 54 components. The bimetallic strip temperature gauge in Fig. 1 is one of the devices. The most complex one, a car distributor system, is shown in Fig. 7. The function of the distributor is to ensure that the spark plugs in the piston chamber fire in sequence at the right time. It works as follows: as the cam rotates, it opens the contact breaker, causing the current in the primary windings to drop rapidly (the condenser prevents a spark from jumping across the contact breaker). The rapid change in current in the primary winding causes a large induced electromotive force in the secondary winding. At the same time, the distributor rotor connects the secondary winding to one of the spark plugs (the rightmost spark plug in the figure). The large induced electromotive force causes a spark to jump across the spark plug. Descriptions of the other devices can be found in [24].

Table 3 summarizes the results of our experiments. The first and second column show the device name and the number of components in each device. This number includes

Table 3
Summary of experimental results

Device name	Number of components	Number of regions	Estimated space	Generated space	Time (sec)
Bimetallic strip temperature gauge	12	1	3.8e16	46	28.4
Bimetallic strip thermostat	10	2	5.4e12	85	43.7
Flexible wire temperature gauge	13	1	2.6e20	78	59.9
Galvanometer temperature gauge	19	1	6.1e31	120	149.8
Electric bell	22	2	6.6e40	117	262.4
Magnetic sizing device	22	1	2.1e51	117	456.0
Carbon pile regulator	26	1	1.5e49	115	262.5
Electromagnetic relay thermostat	30	3	8.7e49	293	472.7
Tachometer	34	1	6.8e58	195	503.6
Car distributor system	54	1	9.9e72	160	352.6

the components created by the program to account for possible component interaction paths. The third column shows the number of operating regions of each device that the model selection program was run on. The fourth column shows the estimated number of consistent and complete device models. These models have at most one model fragment from each assumption class, and a model fragment from each required assumption class. The estimates, derived from the size and organization of the knowledge base, clearly show that a brute-force search for adequate models is totally impractical. The fifth column shows the total number of models actually examined by the program. This is the sum of the number of models examined when the program constructs an initial model and then simplifies it. These low numbers show why our model selection method is practical. The last column shows the actual run time on a Texas Instruments Explorer II workstation.[5]

Table 4 shows the characteristics of the models. The first three columns show the number of model fragments in the most accurate model, in the initial model, and in the (final) adequate model for each device. Multiple entries for a single device correspond to running the program on the different operating regions of that device. The last two columns show the number of equations in the initial and adequate models. Automatically generated device models usually contain a very large number of equations and parameters (e.g., a current for each end of an electrical conductor, and the corresponding Kirchhoff's Current Law equations). We drastically decrease the number of equations by elementary simplifications, e.g., replacing equals by equals. The number of equations reported in Table 4 is the number remaining after such simplifications.

Note that the number of model fragments in the initial model is about half the number in the most accurate model. This demonstrates that the heuristic method is effective in finding an initial model that is significantly simpler than the most accurate model. Note also that, in most cases, the adequate model has on average two-thirds less model fragments than the initial model. This indicates that the heuristic method of finding an initial model is not, by itself, sufficient to find an adequate model, or even a model that

[5] Significantly faster runs have been observed on different machines using different Lisp implementations. For example, the tachometer example has been run in a little over a minute and a half on a Sparc Station 2 under Lucid Lisp version 4.1 [Jon L. White, personal communication].

Table 4
Characteristics of the models

	Number of model fragments			Number of model equations	
Device name	Most accurate model	Initial model	Adequate model	Initial model	Adequate model
Bimetallic strip temperature gauge	75	36	27	33	13
Bimetallic strip thermostat	54	38	14	31	6
	54	39	31	36	11
Flexible link temperature gauge	94	60	25	52	13
Galvanometer temperature gauge	154	98	28	93	17
Electric bell	177	7	6	6	6
	177	108	45	122	35
Magnetic sizing device	202	122	43	126	32
Carbon pile regulator	211	122	51	131	36
Electromagnetic relay thermostat	211	117	31	102	13
	211	119	36	119	30
	211	74	14	73	3
Tachometer	285	170	44	164	30
Car distributor system	348	178	28	192	21

is close to being an adequate model; it must be simplified as described in the previous section.

In conclusion, these results show that the space of complete and consistent device models is too large for brute-force search. The use of causal approximations, as implemented in the model composition algorithm described above, allows the program to systematically explore only a tiny fraction of the search space, making model selection practical. We also conclude that the heuristic technique for finding an initial model is effective in finding significantly simpler than the most accurate model, but is insufficient for finding adequate models. It still needs to be significantly simplified to produce an adequate model.

8. Related work

Our work shares the motivation and builds upon Falkenhainer and Forbus' work on compositional modeling [10]. In their work, they organize their model fragment library by conditioning each model fragment on a set of modeling assumptions. A set of domain-independent and domain-dependent constraints govern the use of these modeling assumptions. Given a user query, they define an adequate model as one that contains all the terms mentioned in the query, and which uses only model fragments that are entailed by a set of assumptions that satisfy the constraints. They construct adequate models using *dynamic constraint satisfaction* [21], and validate them using either qualitative or numerical simulation. If the validation discovers any inconsistencies, the process repeats with the additional information.

While the underlying motivations are similar, our work differs from theirs in a number of significant ways. First, our work focuses on compositional modeling for generating causal explanations of the expected behavior of the device. The expected behavior provides more constraints on model adequacy than a query since not only does it specify the terms that must be included in an adequate model, but it also specifies the causal relations required between parameters. This additional constraint leads to the expected behavior being the central determinant of model adequacy, thereby diminishing the importance of the modeling constraints. Hence, we are able to use less expressive constraints, avoiding the intractability inherent in constraint satisfaction.

Second, we use an order of magnitude reasoning technique for behavior generation, rather than qualitative or numerical simulation. Order of magnitude reasoning is at the right level of abstraction for this application: it doesn't require exact parameter values like numerical methods, while being better than qualitative methods in discriminating between different models. Furthermore, our model selection algorithm incorporates behavior generation correctly, ensuring that behavioral constraints are evaluated against the (relevant part of the) predictions of the most accurate device model. On the other hand, Falkenhainer and Forbus validate a model using behavior generated from the same model. The results of such a validation are questionable if this model incorporates any approximations: while all constraints may be satisfied by the approximate behavior, they may not be satisfied by the behavior predicted by the most accurate model.

Third, we presented a richer, more refined organization of the model fragment library that facilitates knowledge base construction and maintenance and supports focused generation of device models. This organization allows the model selection algorithm to only instantiate those model fragments that are actually used in the search process, instead of instantiating the complete domain theory for the scenario description, as in Falkenhainer and Forbus' algorithm.

Nayak introduced and rigorously analyzed the theoretical basis of causal approximations in [25]. The model composition algorithm developed there simplifies the most accurate model. We built upon this work and extended it in four ways. First, we showed how practical class level descriptions of model fragments should be represented and organized within a model fragment library. Second, we extended the model selection algorithm by including behavior generation using order of magnitude reasoning. This allowed us to ensure that all significant phenomena are included in adequate models. Third, we introduced the component interaction heuristic, which allowed us to find an initial model that is significantly simpler than the most accurate model. Fourth, we tested our implementation on a variety of examples, providing empirical validation of the theoretical claims of [25].

The work on *graphs of models* [1] discusses a technique for selecting models of acceptable accuracy. A graph of models is a graph in which the nodes are models and the edges are assumptions that have to be changed in moving from one model to another. A model in this graph has acceptable accuracy if its predictions are free of conflicts, which are detected either empirically or internally. Empirical conflicts are detected by experimentally verifying a model's predictions, while internal conflicts are detected by checking the model's predictions against a set of consistency rules that capture the model's assumptions. When a conflict is detected, a set of domain-dependent *parameter change* rules help to select a more accurate model, and the above process is repeated.

Analysis begins with the simplest model in the graph of models, and terminates when an accurate enough model has been found. Weld [32] extends this work by introducing *fitting approximations* that form the basis for a domain-independent method for switching to more accurate models.

The main difference between the graphs of models approach and ours is that the graphs of models approach explicitly represents the space of models, while we represent the space of models implicitly using model fragments. Our method leads to greater flexibility in tailoring models to specific situations and provides a compact representation of exponentially many models. To get comparable flexibility in the graphs of models approach is clearly impractical. An advantage of the graphs of models approach is that it identifies well-understood models and can associate with them more efficient specialized problem solvers instead of a general purpose problem solver that is applicable to all models.

The consistency rules used to verify a model's predictions are similar to our behavioral preconditions and coherence constraints. However, we do not validate a model's predictions empirically, and we have not explicitly addressed the problem of switching to a more accurate model when an empirical conflict arises. Our techniques are best viewed as providing an intelligent method for selecting an initial model. Since they always start the analysis with the simplest model, making no effort to identify a better starting model, our techniques are complementary to theirs: select an initial model using our technique, and do model switching using theirs.

Williams' work on producing *critical abstractions* [36] shares our motivations for finding adequate models—we are both striving to find parsimonious descriptions of how a device works. A critical abstraction is a parsimonious description of a device relative to a set of questions. Given a device model, he constructs a critical abstraction in three steps: (a) eliminating superfluous interactions; (b) aggregating interactions that are local to a single mechanism using symbolic algebra; and (c) further abstracting the aggregated interactions. Williams' abstraction process is similar to our model simplification procedure. Specifically, his first step, which eliminates superfluous interactions, is similar to our simplification operator that drops irrelevant model fragments. The primary difference between our approaches is one of emphasis: we have focused on the problem of selecting approximations from a prespecified space of possible approximations, while he has focused on finding techniques for automatically abstracting a base model.

Davis' work on model-based diagnosis [6] has been one of the original inspirations for our work. Davis describes a diagnostic method based on tracing paths of causal interactions. He argues that the power of the approach stems not from the specific diagnostic method, but from the model which specifies the allowed paths of causal interaction. He shows that efficient diagnosis, while retaining completeness, can be obtained by initially considering models with only a few paths of interactions, and adding in additional paths when the model fails to account for the symptoms. Our simplicity ordering on models follows Davis' diagnostic technique: diagnosis starts at an adequate model, with successively more complex models being used if a model is unable to account for the symptoms. The use of causal approximations ensures that using more complex models will add new paths of causal interaction. We have used Davis' definition of component adjacency (two components are adjacent if they can interact with each other by some means). In particular, the component interaction heuristic is closely related to the notion of adjacency: adjacent components must have compatible models.

Reasoning about accuracy is a key aspect of generating adequate device models: a model must be sufficiently accurate to be useful. Recent work has investigated the issues involved in selecting models of acceptable accuracy [8,9,22,29,32,37]. In this paper we have not developed sophisticated techniques for reasoning about model accuracy. A model is deemed to be accurate enough if it satisfies all the behavioral preconditions and coherence constraints, with different levels of accuracy corresponding to different settings of the thresholds. However, our system does not reason about the settings of the thresholds, whose values are part of the input.

In other related work, Liu and Farley present a query-driven method for selecting and shifting between macroscopic and microscopic domain theories [19]. The selection and shift of ontologies is driven by a set of *ontological choice rules*. Iwasaki and Levy show how relevance reasoning can be used for efficiently selecting model fragments for simulation [16]. The efficiency of their algorithm is also based on the use of causal approximations. The primary difference is that they replace our component interaction heuristic with backward chaining along causal influences. Rickel and Porter [27] present an algorithm for selecting appropriate system boundaries for answering prediction questions. The primary difference is that instead of using behavioral constraints to determine the significance of phenomena, they use time scale information.

9. Conclusion and future work

We have presented an efficient model selection algorithm for constructing device models that causally explain how a device functions. Given a structural description of the device and an expected behavior, the algorithm produces a parsimonious device model that causally explains the expected behavior by first constructing an initial model and then simplifying it. The algorithm uses a model fragment library containing class level descriptions of model fragments and their relationships organized for efficient retrieval. It uses the expected behavior and the structural and behavioral contexts of the device to provide task focus and constraints that define model adequacy and restrict the search for adequate models. It uses order of magnitude reasoning to predict device behavior, and component interaction heuristics to produce simpler initial models. By requiring all model fragment approximations to be causal approximations, we guarantee that the number of device models searched by the algorithm is linear in the number of model fragments in the library. We have implemented the algorithm and have successfully used it to produce adequate models and causal explanations for a variety of electromechanical devices from several engineering encyclopedias.

We contemplate several extensions. The model composition algorithm can be easily adapted for incremental model revision and modification to avoid having to construct a new model from scratch in each situation. Incremental model revision is necessary to answer a series of queries, to account for more causal relationships, or to refine an existing model. Instead of starting with an empty model, the model composition algorithm can start with a base model, validating and modifying it as described in steps 1–6 (Fig. 6).

We currently restrict expected behavior descriptions, which have proved to be a useful characterization of device function, to causal relations between parameters. Causal relations are both expressive and can be efficiently checked using causal ordering algorithms. More expressive languages, including additional aspects of function [5], could allow us to represent a wider range of expected behaviors. However, the languages must be tractable for otherwise they compromise the effectiveness of selecting adequate device models, and hence of problem solving.

Accuracy is a very important characteristic of adequate device models. Currently, we do not allow the user to directly specify the desired accuracy of the model, for example by specifying tolerances on certain parameters; we only allow an indirect specification via thresholds in behavioral constraints. Incorporating direct accuracy specifications requires that we develop techniques for finding models that guarantee that predictions will lie within the specified tolerances.

Modeling devices with multiple operating regions is another important capability. Many devices go through multiple operating regions during the course of their normal operations. Different operating regions can have different characteristics, requiring the use of different models. We are currently investigating how to generalize the single-region model composition algorithm to handle multiple operating regions. This will require (a) generalizing the present order of magnitude reasoning technique to allow temporal simulation; and (b) techniques for inferring the expected behavior of each operating region, given the overall expected behavior of the device.

We believe that the compositional modeling framework can be effectively applied to a variety of tasks. Most prominently, we see its use in producing adequate device models for fault diagnosis and monitoring, where recent research has shown an emerging understanding of model adequacy [6, 14]. We believe that the techniques developed in this paper will prove valuable in developing methods for building adequate models for diagnosis and monitoring.

Acknowledgements

We would like to thank Edward Feigenbaum, Richard Fikes, Brian Falkenhainer, Renate Fruchter, Andy Golding, Nita Goyal, Yumi Iwasaki, Rich Keller, Alon Levy, Rajan Ramaswamy, Rich Washington, Dan Weld, and Michael Wolverton for useful discussions and for comments on earlier drafts. Sanjaya Addanki collaborated in the initial phase of this project. We also thank the anonymous reviewers of *Artificial Intelligence* for suggesting many improvements to the original manuscript.

Pandurang Nayak was supported by an IBM Graduate Technical Fellowship. Additional support for this research was provided by the Defense Advanced Research Projects Agency under NASA Grant NAG 2-581 (under ARPA order number 6822), by NASA under NASA Grant NCC 2-537, and by IBM under agreement number 14780042.

References

[1] S. Addanki, R. Cremonini and J. S. Penberthy, Graphs of models, *Artif. Intell.* **51** (1991) 145-177.

[2] I.I. Artobolevsky, *Mechanisms in Modern Engineering Design* **5** (Mir Publishers, Moscow, 1980).

[3] D. Bobrow, ed., *Qualitative Reasoning about Physical Systems* (North-Holland, Amsterdam, 1984).

[4] J.S. Brown, R.R. Burton and J. de Kleer, Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III, in: D. Sleeman and J.S. Brown, eds., *Intelligent Tutoring Systems* (Academic Press, New York, 1982) 227-282.

[5] B. Chandrasekaran, ed., Special Issue on Functional Reasoning, *IEEE Expert* **6** (2) (1991).

[6] R. Davis, Diagnostic reasoning based on structure and behavior, *Artif. Intell.* **24** (1984) 347-410.

[7] J. de Kleer and J.S. Brown, A qualitative physics based on confluences, *Artif. Intell.* **24** (1984) 7-83.

[8] T. Ellman, J. Keane and M. Schwabacher, Intelligent model selection for hillclimbing search in computer-aided design, in: *Proceedings AAAI-93*, Washington, DC (1993) 594-599.

[9] B. Falkenhainer, Ideal physical systems, in: *Proceedings AAAI-93*, Washington, DC (1993) 600-605.

[10] B. Falkenhainer and K.D. Forbus, Compositional modeling: finding the right model for the job, *Artif. Intell.* **51** (1991) 95-143.

[11] K.D. Forbus, Qualitative process theory, *Artif. Intell.* **24** (1984) 85-168.

[12] K.D. Forbus and B. Falkenhainer, Self-explanatory simulations: an integration of qualitative and quantitative knowledge, in: *Proceedings AAAI-90*, Boston, MA (1990) 380-387.

[13] K.D. Forbus and A. Stevens, Using qualitative simulation to generate explanations, in: *Proceedings Third Annual Conference of the Cognitive Science Society*, Berkeley, CA (1981) 219-221.

[14] W.C. Hamscher, Modeling digital circuits for troubleshooting, *Artif. Intell.* **51** (1991) 223-271.

[15] J.R. Hobbs, Granularity, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 432-435.

[16] Y. Iwasaki and A.Y. Levy, Automated model selection for simulation, in: *Proceedings AAAI-94*, Seattle, WA (1994).

[17] Y. Iwasaki and H.A. Simon, Causality in device behavior, *Artif. Intell.* **29** (1986) 3-32.

[18] B.J. Kuipers, Qualitative simulation, *Artif. Intell.* **29** (1986) 289-338.

[19] Z.-Y. Liu and A.M. Farley, Shifting ontological perspective in reasoning about physical systems, in: *Proceedings AAAI-90*, Boston, MA (1990) 395-400.

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

[21] S. Mittal and B. Falkenhainer, Dynamic constraint satisfaction, in: *Proceedings AAAI-90*, Boston, MA (1990) 25-32.

[22] P.P. Nayak, Validating approximate equilibrium models, in: *Proceedings 1991 Model-Based Reasoning Workshop* (1991).

[23] P.P. Nayak, Order of magnitude reasoning using logarithms, in: B. Nebel, C. Rich and W. Swartout, eds., *Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (Morgan Kaufmann, San Mateo, CA, 1992).

[24] P.P. Nayak, Automated modeling of physical systems, Department of Computer Science, Stanford University, Stanford, CA (1992).

[25] P.P. Nayak, Causal approximations, *Artif. Intell.* **70** (1994) 277-334.

[26] R.S. Patil, P. Szolovits and W.B. Schwartz, Causal understanding of patient illness in medical diagnosis, in: *Proceedings IJCAI-81*, Vancouver, BC (1981) 893-899.

[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 (1994) 1191-1198.

[28] D. Serrano and D.C. Gossard, Constraint management in conceptual design, in: D. Sriram and R.A. Adey, eds., *Knowledge Based Expert Systems in Engineering: Planning and Design* (Computational Mechanics Publications, 1987) 211-224.

[29] M. Shirley and B. Falkenhainer, Explicit reasoning about accuracy for approximating physical systems, in: *Working Notes of the Automatic Generation of Approximations and Abstractions Workshop* (1990) 153-162.

[30] H.A. Simon, On the definition of the causal relation, *J. Philos.* **49** (1952) 517-528.

[31] C. van Amerongen, *The Way Things Work* (Simon and Schuster, New York, 1967).

[32] D.S. Weld, Reasoning about model accuracy, *Artif. Intell.* **56** (1992) 255-300.

[33] D.S. Weld and J. de Kleer, eds., *Readings in Qualitative Reasoning about Physical Systems* (Morgan Kaufmann, San Mateo, CA, 1990).
[34] B.C. Williams, Qualitative analysis of MOS circuits, *Artif. Intell.* **24** (1984) 281–346.
[35] B.C. Williams, Interaction-based invention: designing novel devices from first principles, in: *Proceedings AAAI-90*, Boston, MA (1990) 349–356.
[36] B.C. Williams, Critical abstraction: generating simplest models for causal explanation, in: *Proceedings Fifth International Workshop on Qualitative Reasoning about Physical Systems*, Austin, TX (1991).
[37] B.C. Williams and O. Raiman, Decompositional modeling through caricatural reasoning, in: *Proceedings AAAI-94*, Seattle, WA (1994) 1199–1204.