Automated Model Selection using Context-Dependent Behaviors

P. Pandurang Nayak
Knowledge Systems Laboratory
Stanford University
701 Welch Road, Bldg. C, Palo Alto, CA 94304
nayak@cs.stanford.edu

Leo Joskowicz and **Sanjaya Addanki**
IBM T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
josko@ibm.com and addanki@ibm.com

Abstract

Effective problem-solving about complex engineered devices requires device models that are both adequate for the problem and computationally efficient. Producing such models requires identifying the relevant device features and determining applicable simplifications. This paper presents a method for automatically constructing a device model by selecting an appropriate model for each of the device's components using the context in which it operates. We introduce *context-dependent behaviors* (CDBs), a frame-like component behavior model representation for encapsulating contextual modeling constraints. We show how CDBs are used in the model selection process by exploiting constraints from three sources: the structural and behavioral contexts of the components, and the expected behavior of the device. We describe an implemented program for model selection. The inputs are the structure of the device—the components of the device and structural relations between them—the expected device behavior, and a library of CDBs. The output is a set of component CDBs forming a structurally and behaviorally consistent device model that achieves the expected behavior. We demonstrate the program on a temperature gauge.

1 Introduction

Effective problem-solving about complex engineered devices requires device models that are adequate for the problem and computationally efficient. Producing such models requires identifying the relevant device features and determining applicable simplifications. In most existing applications, the user is required to construct the device model appropriate for the task. Constructing models for complex devices with a large space of possible models is a difficult, error-prone, and time-consuming activity requiring skilled and experienced engineers. Automating the model construction process overcomes these drawbacks and provides future intelligent programs with a useful modeling tool.

Model-based reasoning systems construct a device model by composing models of the device components. These systems currently have a single model for each component, thus limiting both the modeling scope and the problems that can be solved. Allowing multiple component models adds flexibility and extends the scope. Producing the simplest, adequate device model consists of selecting component models that are mutually compatible, globally consistent, and incorporate appropriate simplifying assumptions.

The key idea underlying our research is that adequate component models are determined by the context in which they operate. For example, a metallic pipe is modeled either as a physical support or as an electrical conductor depending on whether it supports a water tank or connects a battery to a light bulb. Further, the metallic pipe is modeled either as a rigid or deformable support depending on the strength of the pipe and the weight of the tank. Finally, if the focus is on the pipe's support behavior, its other behaviors (e.g., as a flow channel) are irrelevant and are not modeled. Modeling the pipe as a combination of all its possible models is impractical because it yields overly complicated and intractable models.

We have identified three types of contexts that provide modeling constraints to the model selection process: the structural and behavioral contexts of the components, and the expected behavior of the device. The structural context of a component consists of its physical properties and the components to which it is connected. Structural constraints are modeling constraints on the structural context of a component. For example, if a metallic pipe is connected to a battery, then it is modeled as an electrical conductor. The behavioral context of a component consists of its behavior and the behavior of related components. Behavioral constraints are modeling constraints on the behavioral context of the component. For example, a metallic pipe acted upon by a strong transverse force must be modeled as a deformable body. Expected behaviors are abstract descriptions of device behavior, and are provided by the user. Expected behavior constraints are determined by the device's expected behavior. For example, if the input/output behavior of an overhead flush specifies water flow, the pipe supporting the tank is modeled as a flow channel.

This paper presents a method for constructing simple and adequate device models by selecting appropriate models for each of the device's components. Our method

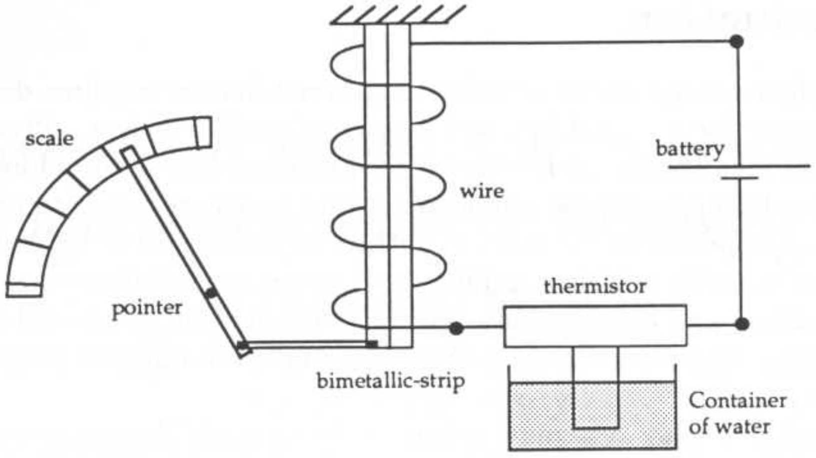

Figure 1: A temperature gauge

exploits the modeling constraints from the structural and behavioral contexts of the components, and the expected behavior of the device. We introduce *context-dependent behaviors* (CDBs), a frame-like component behavior model representation for encapsulating contextual modeling constraints. We describe an implemented program that uses CDBs in the model selection process. The inputs are the structure of the device—the components of the device and structural relations between them—the expected device behavior, and a library of CDBs. The output is a set of component CDBs forming a structurally and behaviorally consistent device model that minimally achieves the expected behavior. We demonstrate the program on a temperature gauge.

2 Example: a temperature gauge

This section presents an example of a device with multiple models for its components, and defines the properties of a good model. Figure 1 shows the schematic of a temperature gauge, consisting 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 temperature gauge works as follows: the thermistor senses the water temperature. An increase in the thermistor's temperature causes its resistance to decrease, causing the current flow in the circuit to increase. This current increase increases the temperature of the wire, thereby increasing the bimetallic strip's temperature. As a consequence, the bimetallic strip bends, causing the pointer to deflect along the scale.

To model the temperature gauge, we use a component model library that contains multiple models for each component. Figure 2 shows part of the wire's space of possible models. For example, the wire can be modeled as an `electrical-conductor`,

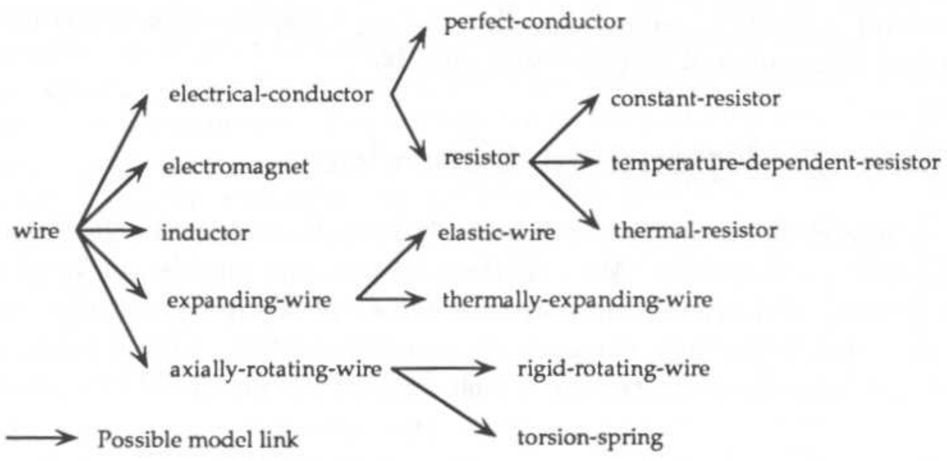

Figure 2: The possible models of a wire.

which can be a `perfect-conductor` or a `resistor`. The `resistor` can be modeled as a `constant-resistor`, a `temperature-dependent-resistor`, or a `thermal-resistor` (which models the heat generated in the resistor). The bimetallic strip can be modeled as a `thermal-bimetallic-strip`, which models the strip's bending due to temperature changes, or as any of the electrical conductor models. The battery can be modeled as a `voltage-source`, which can be either a `constant-voltage-source` or a `variable-voltage-source` (a voltage source with an internal resistance). The thermistor can be modeled as a `thermal-thermistor`, which relates its resistance to its temperature, or as a `thermal-resistor`. The pointer assembly can be modeled as a `rotating-pointer`, which relates the pointer's angular position to the scale reading. All the components can also be modeled as `physical-things` with various thermal, mass, and motion models.

The simplest model that explains the workings of the temperature gauge models the wire both as a `thermal-resistor` and a `constant-resistor`, the bimetallic strip as a `thermal-bimetallic-strip`, the pointer as a `rotating-pointer`, the battery as a `constant-voltage-source`, and the thermistor as a `thermal-thermistor`.

This model satisfies the two important properties of a good model: adequacy and simplicity. Adequacy guarantees that the model correctly captures the device's behavior. For example, ignoring the thermal properties of the wire (modeling it only as a `constant-resistor`) fails to account for the bimetallic strip bending and consequently the pointer's displacement. If the pointer does not move, the device does not measure temperature changes any more. Simplicity guarantees that the model captures only the physical phenomena necessary for explaining the device's behavior. For example, modeling the wire's magnetic and kinematic properties in addition to its thermal and electrical properties produces a consistent but needlessly complicated model with respect to its main function of measuring temperature changes. Selecting an appropriate subset of component models, from a space of possible models, guarantees both properties. In the following, we show how constraints from the structural and behavioral contexts of components and the expected behavior of the device are used to select adequate and simple device models.

3 Context-dependent behaviors

Component models in our system are encapsulations of component behavior and contextual modeling constraints. We call these component models *context-dependent behaviors* (CDBs). Behavioral information is represented with qualitative or quantitative time-varying equations. Contextual modeling constraints are represented with structural and behavioral constraints (which are described in detail in sections 4 and 5).

```
(defcdb resistor (electrical-conductor)
   :
   ;;; Slots of instances of resistor
   ((resistance
       :range resistance-parameter
       :documentation "The resistor's resistance"))
   ;;; Values for slots of resistor
   ((equations
       (= (voltage-difference ?object)
          (* (resistance ?object)
             (terminal-current (electrical-terminal-one ?object))))
       (> (resistance ?object) 0))
    (possible-models constant-resistor
                     temperature-dependent-resistor
                     thermal-resistor)
    (possible-model-of electrical-conductor)
    (contradictory-models perfect-conductor)
    (default-model constant-resistor)
    (structural-constraints)
    (behavioral-constraints
       (implies
          (> (* (terminal-current (electrical-terminal-one ?object))
                (terminal-current (electrical-terminal-one ?object))
                (resistance ?object))
             (electrical-power-dissipation-threshold ?object))
          (model-as ?object thermal-resistor)))))
```

Figure 3: The resistor CDB.

CDBs are represented as frames that inherit properties to their instances. A component is modeled as a CDB by making it an instance of the corresponding frame. As in other frame systems, CDBs are organized into a generalization hierarchy representing the "subset-of" relation between CDBs. CDBs are also organized into a "possible models" hierarchy (see for example figure 2). The possible models of a CDB are the set of CDBs that can be used to model instances of that CDB. The "subset-of" and "possible-models" relations between CDBs may overlap , but are not identical. For example, `resistor` is both a specialization and a possible model of `electrical-conductor`, since every `resistor` is an `electrical-conductor` and each `electrical-conductor` can be modeled as a `resistor`. However, `thermal-thermistor` is a specialization, but not a possible model, of `resistor`, since not every `resistor` can be modeled as a `thermal-thermistor`. Similarly, `electrical-conductor` is a possible model, but not a specialization, of `wire`, since not every `electrical-conductor` is a `wire`.

Figure 3 shows part of the definition of the `resistor` CDB.[1] It is a specialization of the `electrical-conductor` CDB, and it defines the `resistance` parameter for its instances. The `equations` clause describes behavior with equations relating parameters defined for instances of the CDB. The `possible-models` and `possible-model-of` clauses describe the CDB's position in the "possible models" hierarchy. The `contradictory-models` clause identifies models with mutually contradictory assumptions. The `default-model` clause identifies a CDB that must be added to each instance model unless the instance model contradicts the default CDB. For example, a component modeled as a `resistor` should also be modeled as a `constant-resistor`, unless it is also modeled as a `temperature-dependent-resistor` (which contradicts `constant-resistor`).

The `structural-constraints` and `behavioral-constraints` clauses define the CDBs's contextual modeling constraints and are stated in a first-order constraint language. Implication constraints with a `model-as` literal (or a conjunction of `model-as` literals) in the consequent are called `model-as` constraints. A `model-as` constraint is satisfied when the consequent `model-as` literals are satisfied for every variable binding that satisfies the antecedents. A `model-as` literal is satisfied when its first argument is being modeled as an instance of its second argument. Constraints that are not `model-as` constraints are called *general* constraints.

Several CDBs, describing different aspects of a component's behavior, can be combined to produce a component model. For example, a model for a wire can consist of both the `electrical-conductor` CDB and the `electromagnet` CDB when both the wire's electrical and magnetic properties must be modeled. The combined CDBs must be mutually consistent. Combining CDBs supports the modeling of function sharing.

The relationship between components and CDBs is a many-to-many mapping: a single component can be modeled by different CDBs, and a single CDB can model different components. For example, a wire can be modeled by an `electrical- conductor`

[1]Symbols starting with "?" are variables. The variable "?object" is bound to the CDB instance under consideration.

CDB or an `electromagnet` CDB. The `electrical-conductor` CDB can be used to model a wire, a metallic pipe, or the chassis of a car. This many-to-many relation between components and CDBs gives great modeling flexibility for different reasoning tasks. For example, device analysis consists of finding the appropriate CDBs for a given set of components. Device design consists of finding components for a given set of desired behaviors described as CDBs.

In the following sections, we describe in detail the structural and behavioral constraints associated with CDBs.

4 Structural context

The structural context of a component consists of its physical properties (e.g., its shape, mass, and material composition), the structural relations that it participates in, and the components to which it is related by these structural relations. Structural relations are used to describe the structure of a device and include relations such as `connected-to` (indicating that two component terminals are connected), `coiled-around` (indicating that a wire is coiled around a component), and `meshed` (indicating that a pair of gears mesh with each other).

Structural constraints in a CDB are general and `model-as` constraints on the structural context of a component, that must be satisfied if the component is to be modeled by that CDB. For example, the general structural constraint:

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

in the `electrical-conductor` CDB indicates that a component must be metallic for it to be modeled as an `electrical-conductor`. General structural constraints are similar to process preconditions in QP theory [5]. However, unlike process preconditions, these constraints are not used to instantiate CDBs. Hence, the above constraint does not require that every metallic object be modeled as an `electrical-conductor`.

`Model-as` structural constraints are used to enforce the selection of compatible CDBs for structurally related device components. Compatible CDBs allow the structurally related components to interact with each other. For example, the `model-as` constraint:

```
(implies
   (and (terminals ?object ?term1)
        (voltage-terminal-type ?term1)
        (connected-to ?term1 ?term2))
   (model-as ?term2 voltage-terminal-type))
```

in the electrical-component CDB indicates that if a component is modeled as an electrical- component, then every terminal connected to that component's voltage terminals must be modeled as a voltage terminal. This allows the components corresponding to the connected terminals to interact by sharing the voltages at the connected terminals. In addition to being hand-crafted, structural constraints can be derived automatically from the part of the domain theory that specifies component interactions.

5 Behavioral context

The behavioral context of a component consists of its behavior and the behavior of related device components. The behavior of a component is the values, and variations over time of the values, of parameters used to model the component. A component's behavioral context can provide modeling information not explicitly available in the structural context. Behavior generation makes the information implicit in equations become explicit. Consider, for example, a piston moving inside a cylinder. If the piston is reciprocating at a high frequency, as in a car engine, the friction and the heat generated by the piston should be included in the piston-cylinder assembly model. However, if the piston is reciprocating at a low frequency, as in a hand-held bicycle pump, friction and heat can be ignored. The value of the reciprocation frequency (i.e., the behavior) of the piston determines the choice of models.

Behavioral constraints in a CDB are constraints on the behavioral context of a component that must be satisfied if the component is to be modeled by that CDB. For example, the constraint:

```
(implies
   (> (* (terminal-current (electrical-terminal-one ?object))
         (terminal-current (electrical-terminal-one ?object))
         (resistance ?object))
      (electrical-power-dissipation-threshold ?object))
   (model-as ?object thermal-resistor))
```

in the resistor CDB indicates that if the electrical power dissipation in a resistor exceeds a threshold, then this phenomena should be explicitly modeled by modeling the resistor as a thermal-resistor.

The behavioral context can also be used to select appropriate (possibly approximate) models of physical phenomena. For example, suppose we calculate the current flowing in a wire using a perfect-conductor model for it. The following behavioral constraint in the electrical-conductor CDB:

```
(implies
   (and (wire ?object)
        (> (abs (* (terminal-current (electrical-terminal-one ?object))
                   (/ (* (resistivity ?object)
                         (length ?object))
                      (cross-sectional-area ?object))))
                 (voltage-threshold ?object)))
            (model-as ?object resistor))
```

says that if the voltage drop across the wire exceeds a threshold, the wire should be modeled as a `resistor`, rather than as a `perfect-conductor`.

Setting the values of the various thresholds appropriately is crucial for robust model selection. Thresholds can be either preset or computed dynamically. A threshold of 2300 for Reynolds number, that distinguishes laminar flow from turbulent flow, is an example of a preset threshold. Other thresholds can be preset by an engineer from common practice. Thresholds can be set dynamically based on the evolving device model and knowledge of acceptable tolerances on certain parameters (see [11, 9] for some initial work in this area).

6 Expected behavior

The expected behavior of a device is an abstract, possibly incomplete description of *what* the device does (but not *how* it does it). It identifies the important aspects of the device behavior and determines the appropriate device models. We use expected behaviors to capture, in part, what is commonly referred to as the *function* of a device. For example, stating that the device in figure 1 is a temperature gauge indicates that (a) the device model must focus on temperature changes of the thermistor, and that (b) the device model must account for a change in the thermistor temperature causing a change in the angular position of the pointer. The most common expected behavior descriptions are input/output descriptions of the device's behavior.

Knowledge of the expected behavior 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, device names, such as light bulb, vacuum cleaner, and disk drive are widely used and all are associated with expected behaviors. Or suppose we want to know if a disk drive can be used as a door stop. The expected behavior—to stop the door from shutting—suggests that the disk drive model should focus on its kinematic and dynamic properties as an object, not its information retrieval properties! Expected behaviors are an essential component of a device description and play an important role in focusing the model selection process. Without it, all consistent device models are equally plausible: the disk drive as an information retrieval device, a heating device, or a door stop.

Expected behaviors provide two types of constraints: which component parameters *must* appear in the device model, and what are the relations between them.

We specify expected behavior with causal, qualitative or quantitative, equations. For example, the temperature gauge's expected behavior is:

```
(Qprop+ (angular-position pointer) (temperature thermistor))
```

This expected behavior provides the following modeling constraints: the CDB chosen to model the thermistor must have a temperature parameter; the CDB chosen to model the pointer must have an angular position parameter; the pointer's angular position is qualitatively proportional to the thermistor's temperature; and the temperature change causes the angular displacement.

A device model satisfies the constraints from the expected behavior when (a) it includes all the parameters specified in the expected behavior; and (b) the equations of the device model subsume the equations of the expected behavior, i.e., the device model achieves the expected behavior.

7 Modeling algorithm

In this section we describe how the structural constraints, the behavioral constraints, and the constraints from the expected behavior are used to select an adequate device model that is as simple as possible. A device model is said to be *adequate* when (a) it is consistent, i.e., no component model includes contradictory CDBs; (b) the structural and behavioral constraints in each component CDB are satisfied; (c) the expected behavior constraints are satisfied; and (d) all applicable default models are included. A device model M_1 is said to be *simpler than* a device model M_2 if the CDBs selected for each component in M_1 is a subset of the CDBs selected for the component in M_2.

The input to the algorithm is a description of the device and an expected behavior. The device description specifies the device's structure—its components and the structural relations between them—and any user-selected CDBs associated with each component. The algorithm proceeds in four steps. The first step augments the initial device description to include all the expected behavior parameters. The second and third steps enforce the structural and behavioral constraints using dynamic constraint satisfaction [8]. The fourth step checks the expected behavior.

7.1 Details of the algorithm

In the first step, the algorithm checks if the initial device model contains all the expected behavior parameters. If a component parameter is missing, the algorithm searches the `possible-models` of that component for a CDB that provides the required parameter. Only the most general such CDB is returned by this search and is added to the component model. The result is a device model that includes all expected behavior parameters. If more than one most general CDB provides a required parameter, the result is a set of device models corresponding to the different augmentations of the initial device model.

For example, the expected behavior of the temperature gauge (see section 6) requires that the `thermistor` be modeled with a `temperature` parameter and the `pointer` with an `angular-position` parameter. Adding the `thermal-thermistor` CDB to the `thermistor` model and the `rotating-pointer` CDB to the `pointer` model satisfies this requirement.

In the second step, the algorithm checks the structural constraints of each device model. If a general constraint is not satisfied, the device model is deemed inadequate, and is removed from further consideration. If a `model-as` constraint is not satisfied, then it means that the device model does not include a required CDB for some component. The algorithm searches the `possible-models` of that component for the most general CDB that meets this requirement and adds it to the component model. If more than one such CDB is found, the device model is extended to a set of device models. This process is repeated until each device model satisfies all the structural constraints.

In our example, since the `thermistor` is an electrical component (a `thermal-thermistor`), a structural constraint requires that components connected to it must be modeled as electrical components. Modeling the `wire` as an `electrical-conductor` and the `battery` as a `voltage-source` meets this requirement. Similarly, the kinematic interaction required between the `pointer` (a `rotating-pointer`) and the `bimetallic-strip` requires that the latter be modeled as a `thermal-bimetallic-strip`. This requires a thermal interaction between the `bimetallic-strip` and the `wire`, and hence the latter is modeled as a `thermal-object`.

In the third step, the algorithm checks the behavioral constraints for each device model by first generating their behaviors. Behavior generation proceeds in three steps: (a) consistent default models are added; (b) equations are generated by instantiating the `equations` in each CDB; and (c) the equations are solved to get parameter values and bounds. We use BOUNDER [10] to solve the equations and to compute bounds on the parameter values. The behavioral constraints are checked using these values and bounds, and device models are rejected or augmented as described above for the structural constraints. Default models are retracted before device models are augmented to avoid contradictions. The algorithm then repeats the structural constraints check, followed by the behavioral constraints check, until the resulting device models satisfy all the structural and behavioral constraints.

In our example, `perfect-conductor` is added as a default model for the `wire` and `constant-voltage-source` is added as a default model for the `battery`. Using this device model, the circuit current is computed. A behavioral constraint determines that the `wire` should be modeled as a `resistor` since the voltage drop across it is significant. The default models are retracted and `resistor` is added to the `wire` model. Since the structural constraints are still satisfied, default models are once again added to the device model (`constant-resistor` for the `wire` and `constant-voltage-source` for the `battery`), and the circuit current is recalculated.

A behavioral constraint now determines that since the heat generated in the `wire` is significant, it should be modeled as a `thermal-resistor`.

The final step of the algorithm determines if the expected behavior equations are subsumed by the device model equations. The complexity of this test depends on the exact nature of the behavioral equations, and is in general either intractable or infeasible. A weaker, yet effective, test is to check if the device model equations enforce the expected behavior's causality. This is done by first computing the causal ordering [7] of the device model parameters, using the device model equations and assuming that parameters with known signs are exogenous.[2] The causal ordering is then used to check if the expected behavior's causality is satisfied. A device model whose expected behavior parameters' are unrelated is augmented with additional component CDBs. We use a *topology of interactions* [15, 16], built out of the `possible-models` of the device components, to propose additional component CDBs that connect the parameters.

In our example the device model does satisfy the causality specified in the expected behavior—a change in temperature of the thermistor causes a change in the angular position of the pointer. Hence this device model is adequate. It is also the simplest possible because the algorithm only adds a CDB to a component model if it is required to satisfy a constraint.

7.2 Implementation

We have constructed a library of fifteen components, including wires, bimetallic strips, bourdon tubes, and contained gases and liquids. Each component has an average of 15 CDBs describing different aspects of its behavior. The modeling algorithm has been implemented and has been tested on a dozen examples including the temperature gauge (figure 1). These devices have between 10,000 and 100,000 combinations of component models, almost none of which are consistent with the contextual constraints. Our program produces adequate models for these devices in under 5 minutes on an Explorer II.

The modeling algorithm has two main limitations. First, it generates and checks only differential behavior from a point of equilibrium and performs no integration (qualitative or quantitative) over time. Second, expected behavior constraints are limited to behaviors described as qualitative proportionalities (`(Qprop+/- q1 q2)`).

8 Related work

Falkenhainer and Forbus [3, 4] select models by *compositional modeling*. Each model is conditioned on a set of simplifying and operating assumptions. Simplifying assumptions capture a model's approximations, perspectives, and granularity. A set of constraints govern the use of simplifying assumptions. These constraints are similar

[2]Hence this causal ordering is useful only for checking causality in differential behaviors.

to our structural constraints. In addition, we have identified a useful source of these constraints—the observation that components must be modeled in a compatible way. Operating assumptions are similar to behavioral constraints, except that the former are restricted to parameter inequalities, while the latter are general first-order and `model-as` constraints. This means that behavioral constraints can be used, not only to validate approximations, but also to select additional phenomena to be modeled (e.g., selecting `thermal-resistor` because the power dissipated in a `resistor` is too high). Falkenhainer and Forbus use a user query to generate an initial set of simplifying assumptions. This is similar to our use of the expected behavior to generate an initial model. However, in addition, the expected behavior provides feedback on the choice of models—an adequate model's equations must subsume the expected behavior.

Both Addanki *et al* [2, 1] and Weld [14] discuss model switching techniques. Addanki *et al* show how conflicts within the currently selected model can be represented by *delta-vectors,* which are the qualitative changes to parameter values that will eliminate the conflict. Domain-dependent *parameter-change rules* are then used to select models that resolve the conflicts. Weld shows how the domain-independent techniques of intra-model comparative analysis [13, 12] are used to select appropriate models when the models can be formalized as *approximations* of one another. These model switching techniques are similar to our behavioral constraints, in that models are rejected/selected based on the behavior predictions of the device model.

Our definition of the expected behavior, as an abstract description of the actual behavior, is similar to Franke's definition of a *scenario* as a time-ordered sequence of partial qualitative states [6]. One difference is that scenarios abstract a behavior by eliminating states and by leaving out parameters from a state. On the other hand, our expected behaviors abstract a behavior by specifying qualitative, quantitative, and causal relations between parameters.

The algorithm used to connect the parameters of the expected behavior, using the topology of interactions, is based on a similar algorithm used by Williams [15, 16] as part of his Ibis design system.

9 Conclusions

Having multiple models for individual components is necessary to account for the different assumptions, perspectives, and purposes that determine the adequate device model. This paper shows how the context in which the device and its components operate provide a powerful guide for the model selection process. We introduced *context-dependent behaviors* (CDBs), a frame-like component behavior model representation for encapsulating contextual modeling constraints. We showed how CDBs are used in the automated selection of device models by exploiting constraints from three sources: the structural and behavioral contexts, and the expected behavior of the device. We tested our ideas with an implementation.

We believe that our modeling paradigm will prove to be useful for a variety of tasks including analysis, and design. As mentioned in section 3, the compositionality of CDBs and the many-to-many relationship between components and CDBs provides great modeling flexibility. We introduced expected behaviors to allow teleological reasoning (section 6). CDBs provide a uniform mechanism to represent and reason about the structure, behavior, and function of a device.

Future work will involve handling a wider range of expected behaviors, including behaviors over time, dynamic setting of thresholds in behavioral constraints, and the automatic generation of structural and behavioral constraints.

Acknowledgement

We would like to thank Michael Wolverton, Alon Levy, Yumi Iwasaki, Andy Golding, and Edward Feigenbaum for many helpful discussions on this work. We would also like to thank Rich Washington, Rajan Ramaswamy, Rich Keller, Nita Goyal, Renate Fruchter, and four anonymous reviewers for useful comments on earlier drafts of this paper. A special thanks to Elisha Sacks for providing us with the use of BOUNDER. 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] Sanjaya Addanki, Roberto Cremonini, and J. Scott Penberthy. Contexts: Dynamic identification of common parameters in distributed analysis of complex devices. In *Proceedings of IJCAI-89*. International Joint Conference on Artificial Intelligence, 1989.

[2] Sanjaya Addanki, Roberto Cremonini, and J. Scott Penberthy. Reasoning about assumptions in graphs of models. In *Proceedings of IJCAI-89*. International Joint Conference on Artificial Intelligence, 1989.

[3] Brian Falkenhainer and Kenneth D. Forbus. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306. American Association for Artificial Intelligence, 1988.

[4] Brian Falkenhainer and Kenneth D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 1991.

[5] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[6] David W. Franke. Representing and acquiring teleological descriptions. In *Proceedings of the 1989 Workshop on Model Based Reasoning*, pages 62–67, August 1989.

[7] Yumi Iwasaki and Herbert A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986.

[8] Sanjay Mittal and Brian Falkenhainer. Dynamic constraint satisfaction. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 25–32. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.

[9] P. Pandurang Nayak. Validating approximate models. Submitted to the AAAI Model-Based Reasoning Workshop, 1991, 1991.

[10] Elisha Sacks. Hierarchical reasoning about inequalities. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 649–654. American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc., July 1987.

[11] Mark Shirley and Brian Falkenhainer. Explicit reasoning about accuracy for approximating physical systems. In *Working Notes of the Automatic Generation of Approximations and Abstractions Workshop*, pages 153–162, July 1990.

[12] Daniel S. Weld. Comparative analysis. *Artificial Intelligence*, 36(3), October 1988.

[13] Daniel S. Weld. Exaggeration. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 291–295. American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc., August 1988.

[14] Daniel S. Weld. Approximation reformulations. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 407–412. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.

[15] Brian C. Williams. *Invention from First Principles via Topologies of Interactions.* PhD thesis, M.I.T., 1989.

[16] Brian C. Williams. Interaction-based invention: Designing novel devices from first principles. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 349–356. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.