Acquisition of Functional Models: Combining Adaptive Modeling and Model Composition

Sambasiva R. Bhatta

Bell Atlantic
500 Westchester Avenue
White Plains, NY 10604, USA.
bhatta@basit.com

Ashok K. Goel

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, USA.
goel@cc.gatech.edu

Abstract

Functional models explicitly represent the functions of a system and enable teleological reasoning. Structure-behavior-function models of a physical device, for example, specify the device functions and also its internal causal behaviors that explain how the device structure achieves the functions. They enable reasoning about the device teleology in tasks such as adaptive design and redesign. An open issue in research on functional models concerns their origin and acquisition. We describe a computational technique that combines the methods of adaptive modeling and model composition. In the integrated method, the model of the new device is acquired by revising the model of a similar device, and, if the new device contains structural elements not in the similar device, then by consolidating the models of the new elements with the model of the similar device. This technique is one part of a general computational theory of conceptual design called *model-based analogy*.

Background, Motivations, and Goals

Functional models explicitly represent the functions of a system to enable teleological reasoning about function-related tasks (Sembugamoorthy and Chandrasekaran 1986). Structure-behavior-function (SBF) models of physical devices, for example, specify the structure, the functions, and the internal causal behaviors that explain how the device structure results in its functions (Goel 1991). In particular, the internal behaviors specify how the functions of the structural elements are hierarchically composed into the functions of the device. SBF models have proved to be quite useful for teleological reasoning about function-related tasks such as variant and adaptive design, design verification, and redesign (Goel, Bhatta, and Stroulia 1997). Since they explicitly represent the functions and use them to organize behavioral knowledge, they help define problem spaces for design adaptation, verification and redesign, and provide access to the knowledge relevant for searching the spaces. They also give rise to a vocabulary for indexing designs.

But the origin, generation and acquisition of functional models remain open issues. Not only are these questions fundamental, but, in addition, their answers are likely to impose additional constraints on the models. We describe a computational strategy for acquiring SBF models that integrates the methods of adaptive modeling and model composition. Given the structure of a new device as input, the former method generates a model for the new device by revising the model of a similar device, while the latter method generates the device model by composing models of the device elements. The new strategy integrates Bylander's (1991) consolidation method of model composition with our method of adaptive modeling (Goel 1991, 1996). In the new strategy, the model of the new device is generated by revising the model of a similar device, and, if the new device contains structural elements not in the similar device, then by consolidating models of the new elements with the model of the similar device. The choice of consolidation as the method of model composition is due to its ontological compatibility with our adaptive-modeling method. The integrated strategy combines the efficiency of adaptive modeling with the generality and power of model composition.

The origin of this work lies in our research on analogy-based innovative design. We have developed a computational theory of innovative design called *model-based analogy* (or MBA) (Bhatta 1995). We have also instantiated and evaluated MBA in an operational computer program called IDEAL. The system solves function $\rightarrow$ structure design problems autonomously, by model-based retrieval and adaptation of design cases, and by model-based transfer of design abstractions over known cases. Depending on its background knowledge relative to a given problem, IDEAL might fail to solve the problem. If and when it fails, the system interacts with an oracle. If the oracle supplies the correct design, then IDEAL generates an SBF model for the new design and stores it in its case memory for potential reuse. This raises the issue of generation and acquisition of SBF models, and leads to the new integrated strategy. Thus the new model-acquisition strategy is one part of the MBA theory, and is instantiated in IDEAL.

In this paper, we use the simplest example from IDEAL to illustrate the new strategy, but the strategy is applicable to a large range of problems as demonstrated by IDEAL. Also, we are not claiming that generating a model for the simple device in this example requires the integrated method, but merely using the example for illustration.

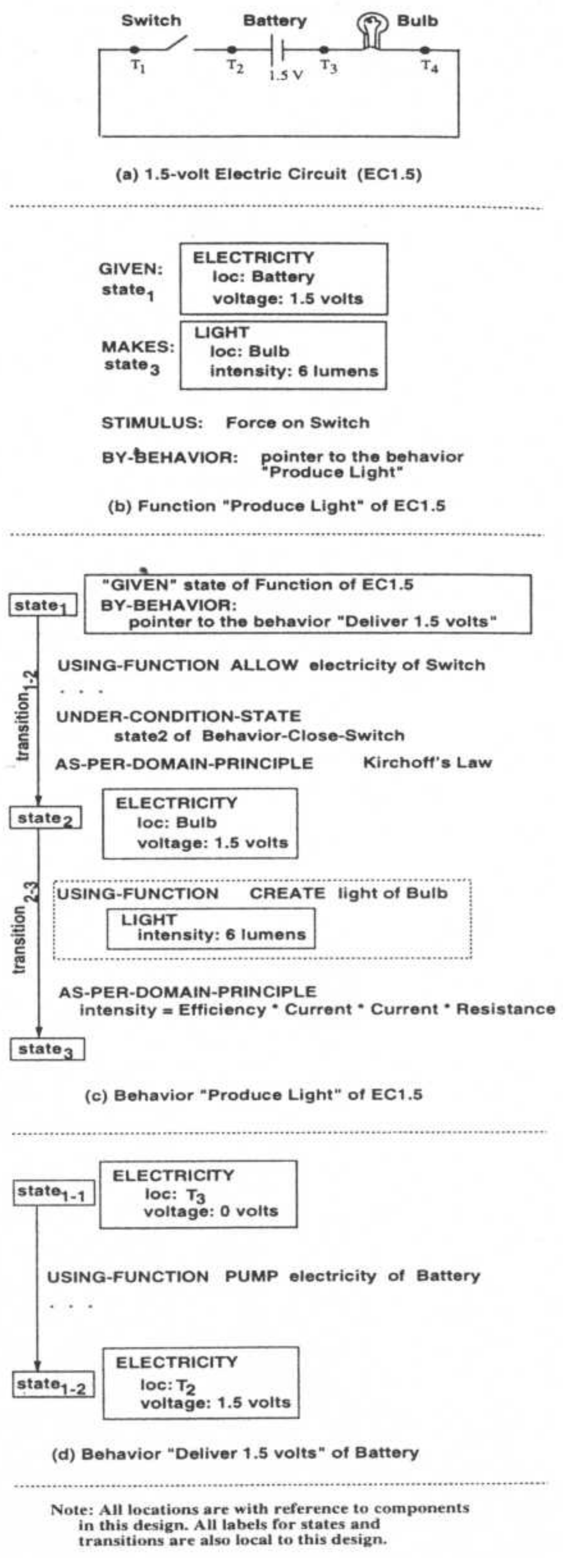
SBF Device Models

In IDEAL, a stored design case specifies (a) the functions delivered by the known design, (b) the structure of the design, and (c) a pointer to the causal behaviors of the design (the SBF model). The design cases are indexed both by functions that the stored designs deliver and by the structural constraints they satisfy. The representations in this work are motivated by the bigger context for this work, namely, model-based analogy, that addresses learning, memory and problem solving in design.

The SBF model of a stored design expresses IDEAL's comprehension of how the device works, i.e., how its structure results in its output behaviors which include its functions. (The output behaviors are abstractions of internal behaviors, and functions are the intended output behaviors.) The SBF language provides conceptual primitives for representing and organizing knowledge of a device including its structure (i.e., the device topology), functions, and internal behaviors. In the SBF ontology, the **structure** of a device is viewed as constituted of *components*, *substances*, and relations among them. Substances have *locations* in reference to the components in the device. They also have *behavioral properties,* such as *voltage* of *electricity,* and corresponding *parameters,* such as *1.5 volts*.

A **function** in the SBF models is a behavioral abstraction and is represented as a schema that specifies the behavioral state the function takes as input, the behavioral state it gives as output, and a pointer to the internal causal behavior of the design that achieves the function. Figure 1(a) illustrates the design of a simple electric circuit which we will refer to as EC1.5. Figure 1(b) illustrates the SBF specification of the function "Produce Light" of EC1.5. Both the input and the output states are represented as *substance schemas.* The input state specifies that electricity at `location` Battery in the topography of the device (Figure 1(a)) has the property `voltage` and the corresponding parameter `1.5 volts`. The output state specifies the property `intensity` and the corresponding parameter `6 lumens` of a different substance, light, at `location` Bulb. In addition, the slot *by-behavior* acts as an index into the causal behavior that achieves the function of producing light.

The internal causal **behaviors** in the SBF model of a device explicitly specify how the functions of structural elements of the device get composed into device functions. They are represented as sequences of states and state-transitions. The annotations on the state transitions express the *causal, structural,* and *functional context* in which the transformation of state variables, such as substance, location, properties, and parameters, can occur. Figure 1(c) shows the causal behavior that explains how electricity in Battery is transformed into light in Bulb. $State_1$ describes the state of electricity at location Battery and so does $state_2$ at location Bulb. $State_3$ however describes the state of light at location Bulb. The annota-



Switch
Battery
Bulb
T1
T2
1.5 V
T3
T4
(a) 1.5-volt Electric Circuit (EC1.5)
GIVEN: state1
ELECTRICITY loc: Battery voltage: 1.5 volts
MAKES: state3
LIGHT loc: Bulb intensity: 6 lumens
STIMULUS: Force on Switch
BY-BEHAVIOR: pointer to the behavior "Produce Light"
(b) Function "Produce Light" of EC1.5
state1
"GIVEN" state of Function of EC1.5
BY-BEHAVIOR: pointer to the behavior "Deliver 1.5 volts"
transition1-2
USING-FUNCTION ALLOW electricity of Switch
. . .
UNDER-CONDITION-STATE state2 of Behavior-Close-Switch
AS-PER-DOMAIN-PRINCIPLE Kirchoff's Law
state2
ELECTRICITY loc: Bulb voltage: 1.5 volts
transition2-3
USING-FUNCTION CREATE light of Bulb
LIGHT intensity: 6 lumens
AS-PER-DOMAIN-PRINCIPLE intensity = Efficiency * Current * Current * Resistance
state3
(c) Behavior "Produce Light" of EC1.5
state1-1
ELECTRICITY loc: T3 voltage: 0 volts
USING-FUNCTION PUMP electricity of Battery
. . .
state1-2
ELECTRICITY loc: T2 voltage: 1.5 volts
(d) Behavior "Deliver 1.5 volts" of Battery
Note: All locations are with reference to components in this design. All labels for states and transitions are also local to this design.

Figure 1: **A Design for A 1.5-volt Electric Circuit (EC1.5)**

tion USING-FUNCTION in $transition_{2-3}$ indicates that the transition occurs due to the primitive function "**create** light" of Bulb. The causal behaviors can be specified at different levels of detail. For instance, $state_1$ is an aggregation of a sequence of several states and transitions at a different level as shown in Figure 1(d).

Generation and Acquisition of SBF Models

MBA is a unified theory of learning, memory and problem solving in design. Therefore, we need to describe the state of memory and of problem solving before we can describe the strategy for acquiring SBF models. In particular, we need to set a context in which IDEAL fails to solve a problem and an oracle presents the correct solution to the system.

Consider the scenario in which IDEAL is presented with a problem of designing a device that can deliver light of intensity 12 lumens. The system's case memory contains the design of EC1.5 shown in Figure 1. It retrieves the design of EC1.5 from the case memory because the given functional specification is similar to the function of EC1.5. It then views the EC1.5 circuit, which delivers only 6 lumens of light, as having failed to deliver the desired 12 lumens of light. It uses the SBF model of EC1.5 to first diagnose the cause for this failure and then to repair it. Thus first it identifies that the cause for the failure of the EC1.5 circuit to deliver 12 lumens of light is that the voltage of 1.5 volts delivered by the battery in the circuit is too low, and determines that 3 volts of electricity is needed to deliver 12 lumens of light. Then IDEAL retrieves and instantiates a generic repair plan (component-replacement) applicable to the current task, and decides to replace the 1.5-volt battery in EC1.5 with a 3-volt battery. Next IDEAL searches its design memory which does not contain a 3-volt battery. At this stage, it abandons the candidate modification of replacing the 1.5-volt battery with a 3-volt battery. It knows of no alternative repair plan applicable to the current task, and can generate no alternative diagnoses. Finally IDEAL determines that it cannot solve the problem given to it, suspends its processing, and informs the oracle of its failure. IDEAL inherits its methods for design retrieval, diagnosis and repair from the KRITIK system, explained in detail in (Goel, Bhatta, and Stroulia 1997).

Now consider the interactions between IDEAL and the oracle. The oracle may provide the system with different kinds of information: (1) the correct design solution along with a complete SBF model; (2) only the correct design solution (i.e., only the structure of the design); or (3) only a partial solution in the context of the specific problem-solving failure (i.e., a localized-structure for the specific adaptation task). In the first case, IDEAL needs to store the new design and its SBF model in its memory for potential reuse. In the second case, it needs to generate an SBF model from the structure before storing the solution in memory. In the third case, it needs to first complete the partial solution.

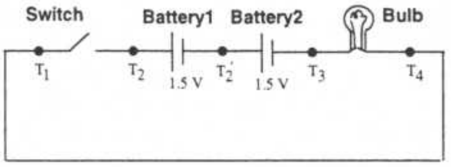


Figure 2: **A 3-volt Electric Circuit (EC3)**

Here we focus on the second case: the oracle specifies only the structure of the desired design, with the function already given in the design problem. Figure 2 schematically shows the design given by the oracle in our running example. In this design, 3 volts of electricity is obtained by connecting two 1.5-volt batteries in series (we refer to this circuit as EC3). Now IDEAL needs to acquire the internal behaviors of EC3, which explain how the design delivers the desired 12 lumens of light.

Integration of Model Adaptation and Model Composition

Given the structural specification of a device, to generate internal behaviors of the device, IDEAL's strategy combines the methods of model revision and model consolidation. It revises the behaviors of the known design by mapping the components in the new structure onto those in the structure of the known design, and by consolidating the behaviors of any additional components in the new structure with the behaviors of the known design. The strategy contains four main steps: (1) transfer of the internal behavior of the known device to the functionally-similar new device to obtain an incomplete behavior for the new device, (2) mapping of components in the structure of the known device onto the components in the structure of the new device, (3) insertion of functions of additional components in the new device in its incomplete internal behavior to obtain a complete behavior, and (4) propagation of new parametric values introduced by the insertion of the additional components through the internal behavior (by applying qualitative relationships among parameters).

Figure 3 partially specifies the structure of EC3 given by the oracle to IDEAL. The task is to generate the internal behaviors of this structure. As a first step, IDEAL makes a copy of the behavior of EC1.5 (Figure 1). It then modifies each behavior segment (i.e., a state-transition-state unit) in the copy by mapping components in the behavior of the known device onto components in the new structure. For component mapping, it uses knowledge of the structural context in the known behavior (which specifies the connections between components) and the functions of the components. For instance, the Bulb in EC1.5 is mapped only onto the Bulb in EC3 (and not onto any other component) because the structural context (i.e., a serial connection with a battery at a particular end and a serial connection with switch at a particular end) and functions of Bulb in EC1.5 are similar to the structural context and functions of Bulb in EC3, respectively. In

STRUCTURE CircuitEC3
components: (Battery1, Battery2, Switch, Bulb)

STRUCTURE Battery1
relations: (SERIALLY-CONNECTED Switch Battery2)
parameters: (voltage 1.5 volts)
functions: (ALLOW electricity)
(PUMP electricity)
connecting-points: (T2' T2)

STRUCTURE Battery2
relations: (SERIALLY-CONNECTED Battery1 Bulb)
parameters: (voltage 1.5 volts)
functions: (ALLOW electricity)
(PUMP electricity)
connecting-points: (T3 T2')

STRUCTURE Switch
relations: ...
STRUCTURE Bulb
relations: ...

Figure 3: **Structure of EC3 in Schema Form**

general, this need not guarantee a unique mapping between the structures of the known and the new designs. For instance, either battery in EC3 can map onto the battery in EC1.5. (Both mappings are equivalent in this case because of symmetry of the connections but this need not be true in general.) IDEAL's strategy works with any of these multiple (apparently valid) mappings; it selects one arbitrarily and proceeds with it. Thus, let us suppose that it maps the battery in EC1.5 onto Battery2 in EC3. This leads to the generation of the behavior for the new structure illustrated in Figure 4. This behavior is incomplete (and incoherent) because the behavior of the additional component (i.e., Battery1 in EC3) is yet to be composed.

In order to revise the incomplete behavior to accommodate any additional components in the new structure, for each additional component IDEAL first needs to determine *where* in the incomplete device behavior to compose the component behavior. That is, it needs to determine the state(s) in the incomplete behavior where it should add the new component behavior. The states in the incomplete behavior specify the connections between components and the *locations* of the connections in the device space, and this knowledge enables IDEAL to identify the state at which to add the behavior of an additional component. Thus IDEAL identifies that the behavior of Battery1 in EC3 needs to be added after the state at location T_2' in the incomplete behavior of EC3 (Figure 4(b)). For each component, its (device-independent) function is specified as a transition between two states (the Given state at the input end and the Makes state at the output end of the component). Based on information about the points at which the component behavior will be added in the incomplete behavior, and the input and output ends of the component, IDEAL first generates the initial and final states of

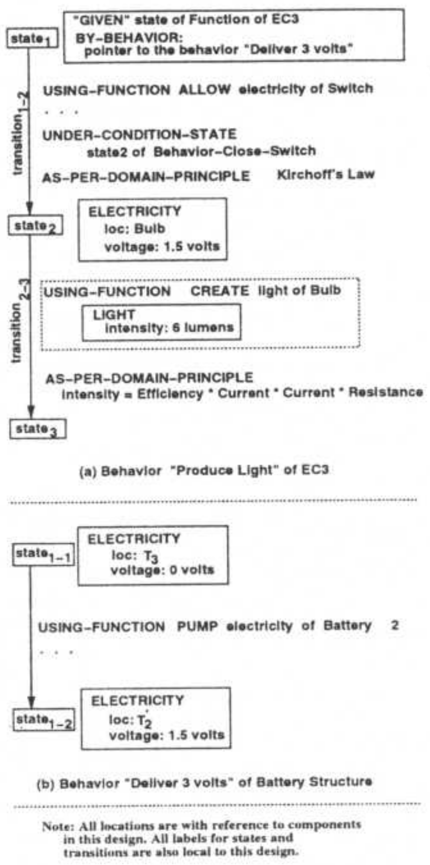

"GIVEN" state of Function of EC3
BY-BEHAVIOR:
pointer to the behavior "Deliver 3 volts"
state1
transition1-2
USING-FUNCTION ALLOW electricity of Switch
. . .
UNDER-CONDITION-STATE
state2 of Behavior-Close-Switch
AS-PER-DOMAIN-PRINCIPLE Kirchoff's Law
state2
ELECTRICITY
loc: Bulb
voltage: 1.5 volts
transition2-3
USING-FUNCTION CREATE light of Bulb
LIGHT
intensity: 6 lumens
AS-PER-DOMAIN-PRINCIPLE
intensity = Efficiency * Current * Current * Resistance
state3
(a) Behavior "Produce Light" of EC3
state1-1
ELECTRICITY
loc: T3
voltage: 0 volts
USING-FUNCTION PUMP electricity of Battery 2
. . .
state1-2
ELECTRICITY
loc: T2'
voltage: 1.5 volts
(b) Behavior "Deliver 3 volts" of Battery Structure
Note: All locations are with reference to components in this design. All labels for states and transitions are also local to this design.

Figure 4: **Incomplete behavior of EC3 in the process of behavior generation**

the component behavior in the incomplete behavior, although some parameters may be instantiated later. Then the function of the component that best describes the transition between these initial and final states is selected to describe the transition between them. (This is needed because a component can have multiple functions, only one of which may be relevant in the current device.) Each of the component functions may also specify qualitative relations between parameters at the input and the output ends. When a function of the component is selected to describe the transition in the incomplete behavior, the corresponding qualitative relations are also selected. These relations are modified to reflect the parameters in the initial and final states of the transition, and added to the specification of the transition. Figure 5 illustrates the transition thus generated for Battery1 in EC3. This transition is inserted in the incomplete behavior after the state at location T_2' (i.e., $state_{1-2}$), and the parameter changes due to this insertion are propagated to the subsequent states and transitions in the incomplete behavior (by simulating the model using the information in the transitions such as qualitative relations). The complete behavior for the given structure of EC3 is illustrated in Figure 6. Thus the integrated method combines adapting a similar device model and consolidat-

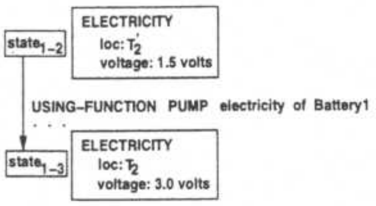


Figure 5: **Behavior of Battery1 in EC3**

ing the models of new components with the device model in order to acquire the SBF model of a new device.

Evaluation

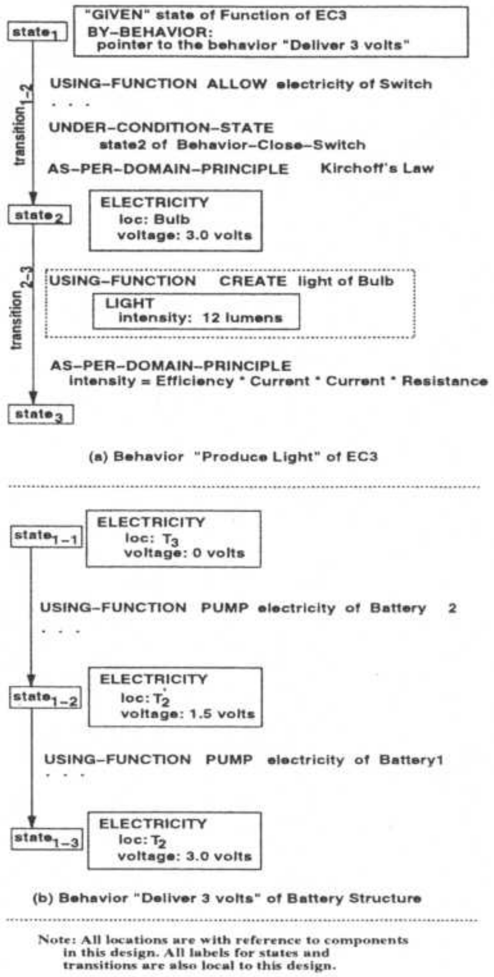
IDEAL provides a testbed for experimenting with the MBA strategy for acquiring SBF models. We evaluated the strategy in the following dimensions (details in (Bhatta 1995)).
(1) Computational Efficacy and Domain Generality: We tested the model-acquisition strategy for a dozen problems in the domains of electric circuits, heat exchangers, and electronic circuits (with operational amplifiers). The hardest example (design of an acid cooler) contains about a dozen components.
(2) Different Interaction Conditions: We tested the model-acquisition strategy under different kinds of interaction conditions between IDEAL and the oracle. They are characterized by the types of information the oracle provides as external feedback upon a problem-solving failure. In particular, we verified that the strategy works when the oracle provides as feedback (a) only the solution (structure) for the design problem, and (b) only the solution (substructure) specific to the local adaptation goal.
(3) Use of the Acquired Models: We tested and found that the device models IDEAL learns are useful for subsequent tasks in the MBA process (e.g, learning design abstractions), and, more importantly, for solving new design problems. This suggests that the strategy leads to the acquisition of correct models.

Discussion

We will use comparison with related research and our plans for future work as vehicles to discuss the contributions and limitations of the integrated strategy for model generation and acquisition.

Related Research

The strategy for generating and acquiring functional models is related to several lines of research in cognitive science, artificial intelligence and computer-aided design. Below we briefly compare it only with similar AI research on qualitative reasoning. We have already indicated the relationship between our SBF device models and Sembugamoorthy and Chandrasekaran's (1986) functional representations. SBF device models are well-defined functional representations: they are based on a specific device ontology that gives rise to a language for representing the behavioral states of a device, which enables principled



state1
"GIVEN" state of Function of EC3
BY-BEHAVIOR:
pointer to the behavior "Deliver 3 volts"
transition1-2
USING-FUNCTION ALLOW electricity of Switch
UNDER-CONDITION-STATE
state2 of Behavior-Close-Switch
AS-PER-DOMAIN-PRINCIPLE Kirchoff's Law
state2
ELECTRICITY
loc: Bulb
voltage: 3.0 volts
transition2-3
USING-FUNCTION CREATE light of Bulb
LIGHT
intensity: 12 lumens
AS-PER-DOMAIN-PRINCIPLE
intensity = Efficiency * Current * Current * Resistance
state3
(a) Behavior "Produce Light" of EC3
state1-1
ELECTRICITY
loc: T3
voltage: 0 volts
USING-FUNCTION PUMP electricity of Battery 2
state1-2
ELECTRICITY
loc: T2
voltage: 1.5 volts
USING-FUNCTION PUMP electricity of Battery1
state1-3
ELECTRICITY
loc: T2
voltage: 3.0 volts
(b) Behavior "Deliver 3 volts" of Battery Structure
Note: All locations are with reference to components in this design. All labels for states and transitions are also local to this design.

Figure 6: **Complete Behaviors of EC3 generated from the given structure**

representation of device functions and behaviors. Umeda et al. (1990) describe FBS device models similar to SBF models. Like our SBF models, their FBS models explicitly represent device functions, and use behaviors to mediate between functions and structure. However, our SBF models are organized in a composition hierarchy, with the behaviors at each level in the hierarchy decomposing the functions of the device at that level into the functions of the subdevices at the next lower level.

Adaptive modeling is a similarity-based strategy for the generation and acquisition of functional models: if a new device is structurally similar to a known device such that their structural differences are limited to parametric values of specific components or one-to-one replacement of a specific component by another component, then it becomes possible to generate the behaviors of the new device by directly transferring and tweaking the behaviors of the known device.

Compositional modeling is a very general and powerful strategy for the generation of behaviors from structure.

CML (Falkenhainer et al. 1994) is a recent language based on a general ontology for supporting compositional modeling. In compositional modeling, the (output) behaviors of a given device are derived by composing the (output) behaviors of its structural components. The behavior composition is governed both by the structural relations among the components and by general rules of composition. The composition process is driven by an iterative application of the compositional rules to increasingly large component assemblies. The process results in a specification of all potential output behaviors of the device. While compositional modeling is one way of achieving model composition, Bylander's consolidation method is another. We have chosen to use the consolidation method for model composition because of its ontological compatibility with SBF models and the adaptive-modeling method.

The integrated strategy combines the efficiency of adaptive modeling with the generality and power of model composition. The similarity-based adaptive-modeling strategy, while efficient, is limited to problems in which the structural differences between the new device and the known device are small, simple, and local. The strategy of model composition, while very general and powerful, can be computationally complex. In the integrated strategy, adaptive modeling localizes model composition, thereby reducing its complexity. Also, adaptive modeling provides top-down guidance to the bottom-up method of model composition, thereby focusing the latter to the actual functions of the device.

Falkenhainer (1990) describes an analogy-based strategy for completing almost-complete behavioral models of physical processes. In his PHINEAS system, Falkenhainer integrates the principles of structure mapping (Gentner 1983) with qualitative process representations (Forbus 1984). The task PHINEAS addresses is learning of a theory of a specific physical behavior by a similarity-driven explanation mechanism, which involves analogical mapping of the explanation of a similar physical situation. The mapping of components in the structure of the known device onto the components in the structure of the new device in our strategy too is based on the principles of structure mapping. However, while PHINEAS assumes that an almost-complete model of the new system is known, our strategy assumes that the new system is sufficiently similar to the known system that the internal behavior of the known system can be directly transferred to generate an initial, though incomplete, behavior of the new system. PHINEAS completes the almost-complete model of the new system by generating behavioral abstractions of the known system and transferring them to the model of the new system. In our strategy, the SBF model of the known device, retrieved from memory, directly provides the functional abstractions of components. These abstractions localize and guide the composition of additional components in the new device structure with its incomplete behavior to obtain a complete SBF model.

Future Work

The current version of the new strategy for generating and acquiring functional models is limited in (at least) two aspects, but we are planning to address these limitations: (1) the structure of the new design can have one or more additional components relative to those in the structure of the known design, but it cannot have fewer components; and (2) the additional components are connected serially with other components corresponding to the components in the known design. We also plan to formally analyze properties of the strategy and the solutions it generates.

References

Bhatta, S. 1995. Model-Based Analogy in Innovative Device Design. Ph.D. diss., College of Computing, Georgia Institute of Technology.

Bylander, T. 1991. A Theory of Consolidation for Reasoning about Devices. *The International Journal of Man-Machine Studies* 35:467–489.

Falkenhainer, B. 1990. A Unified Approach to Explanation and Theory Formation. In J. Shrager and P. Langley (eds.), *Computational Models of Scientific Discovery and Theory Formation*. San Mateo, CA: Morgan Kaufmann.

Falkenhainer, B.; Farquhar, A.; Bobrow, D.; Fikes, R.; Forbus, K.; Gruber, T.; Iwasaki, Y.; and Kuipers, B. 1994. CML: Compositional Modeling Language, Technical Report, KSL-94-16, Knowledge Systems Laboratory, Stanford Univ.

Forbus, K. 1984. Qualitative Process Theory. *Artificial Intelligence* 24:85–168.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science* 7(2):155–170.

Goel, A. 1991. Model Revision: A Theory of Incremental Model Learning. In *Proceedings of the Eighth International Conference on Machine Learning*, 605–609. Los Altos, CA: Morgan Kaufmann.

Goel, A. 1996. Adaptive Modeling. In *Proceedings of the Tenth International Workshop on Qualitative Reasoning*. Stanford Sierra Camp, California.

Goel, A.; Bhatta, S.; and Stroulia, E. 1997. Kritik: An Early Case-Based Design System. In M. Maher and P. Pu (eds.), *Issues in Case-Based Design*. Hillsdale, NJ: Erlbaum.

Sembugamoorthy, V. and Chandrasekaran, B. 1986. Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems. In J. Kolodner and C. Riesbeck (eds.), *Experience, Memory and Reasoning*, 47–73. Hillsdale, NJ: Erlbaum.

Umeda, Y.; Takeda, H.; Tomiyama, T.; and Yoshikawa, H. 1990. Function, Behavior and Structure. In *Proceedings of the Fifth International Conference on Applications of AI in Engineering*, 177–193.