From Design Experiences To Generic Mechanisms: Model-Based Learning in Analogical Design[*]

Sambasiva R. Bhatta and **Ashok K. Goel**
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
{bhatta,goel}@cc.gatech.edu

In the Proceedings of the AID-94 workshop on Machine Learning in Design, Aug. 1994, Lausanne, Switzerland.

Abstract

Analogical reasoning plays an important role in design. In particular, cross-domain analogies appear to be important in innovative and creative design. However, making cross-domain analogies is hard and often requires abstractions common to the source and target domains. Recent work in case-based design suggests that generic mechanisms are one type of abstractions useful in adapting past designs. However, one important yet unexplored issue is where these generic mechanisms come from. We hypothesize that they are acquired incrementally from design experiences in familiar domains by generalization over patterns of regularity. Three important issues in generalization from experiences are what to generalize from an experience, how far to generalize, and what methods to use. In this paper, we describe how structure-behavior-function models of designs in a familiar domain provide the content, and together with the problem-solving context in which learning occurs, also provide the constraints for learning generic mechanisms from design experiences. In particular, we describe the model-based learning method with a scenario of learning of feedback mechanism.

Introduction

Analogical reasoning, it is commonly accepted, plays an important role in design in general, and in innovative and creative design in particular. Analogical reasoning is the process of retrieving knowledge of a familiar problem or situation (called *source analog*) that is relevant to a given problem (called *target*) and transferring that knowledge to solve the current problem. Analogies can be of different types: within-problem, cross-problem but within-domain (in short, within-domain), and cross-domain. Much of the existing work in analogical design was in the framework of case-based reasoning and has focused on within-domain analogies, e.g., the earlier versions of STRUPLE (Maher & Zhao, 1987) and KRITIK (Goel, 1991). That is, they are limited to the retrieval of design cases from the same domain as a given problem, and to making small and simple 'tweaks' to the retrieved design to fit the specifications of the given problem. In contrast, we are interested in cross-domain analogies in design.

Let us first illustrate what we mean by cross-domain analogy. A domain is characterized by the specific structural elements available in it such as pipes, batteries, and diodes. **Cross-domain analogy** involves transferring knowledge from a problem in one domain to another problem in a different domain. For instance, consider the problem of designing a Nitric acid cooler that is required to cool Nitric acid over a large range of temperature. One way in which this can be achieved is by putting together several acid coolers each of which cools acid over a smaller temperature range. A designer may solve this problem by transferring knowledge from a past experience of designing a different kind of device, say, an electric circuit. In particular, the designer may instantiate an abstraction such as the generic mechanism of cascading (replicating a smaller device several times) learned from specific experiences in the domain of electric circuits to design the Nitric acid cooler. Cross-domain analogies are the hardest because of the issue of recognizing the similarity between two problems from two different domains and transferring knowledge between them.

Previous research on analogy in both AI (e.g., Greiner, 1988; Falkenhainer, 1989) and psychology (e.g., Gick & Holyoak, 1983; Catrambone & Holyoak, 1989) has indicated that abstractions shared between source and target domains help in cross-domain analogies. Much of the previous work has only focused on the role of abstractions in retrieving cases (i.e., using abstractions as indices) but not in the transfer of knowledge. Some of the issues of interest then are what might be the content, representation, and organization of these abstractions and how might they be acquired and used for transfer in analogical design.

In some earlier work on case-based design, generic teleological mechanisms (GTMs), such as cascading, feedback, and feedforward, have been proposed as one type of abstract knowledge that designers might use in adapting design cases (Goel, 1989). GTMs are *teleological* because they result

[*]This work has been supported by NSF (research grant IRI-92-10925), ONR (research contract N00014-92-J-1234), Northern Telecom (research gift), and ARPA.

in specific functions and are *generic* because they are case independent. GTMs take as input the functions of a desired design and a known design, and suggest patterned modifications to the structure of the known design that would result in the desired design. We have been exploring the issue of their representation and use in analogical design. Stroulia & Goel (1992) describe how the generic mechanism of cascading indeed is useful in non-routine adaptive design. Another important issue relevant to learning in design is how these generic mechanisms are acquired. Our working hypothesis is that these generic mechanisms can be acquired incrementally from design experiences in familiar domains by generalization over patterns of regularity. In the IDEAL project,[1] we have been exploring these issues and hypotheses in the context of analogical design of physical devices such as simple electric circuits, electronic circuits with operational amplifiers, and heat exchangers.

We have earlier described elsewhere (Bhatta & Goel, 1993) how IDEAL can learn the generic mechanism of cascading from design experiences in analogical design. Since then, we have generalized the model-based learning method in order to reason about devices with "non-linear" behaviors (i.e., those that have cycles (or loops), single cause-multiple effects (i.e., forks) and multiple causes-single effect (i.e., joins), etc.), and learn other classes of mechanisms such as feedback and feedforward. The current version of IDEAL implements the generalized model-based learning method. In this paper, we will describe IDEAL's generalized learning method and illustrate it with a scenario of learning of feedback mechanism from designs in the domain of electronic circuits. The evaluation of the learning of feedback and feedforward mechanisms along the dimension of their utility in facilitating cross-domain transfer is underway.

Issues in Learning by Generalization

Generalization from experiences raises three important issues. First is the issue of relevance—the issue of deciding what to generalize from an experience. In this regard, the existing machine learning methods such as pure induction over design experiences could potentially become complex. We have argued elsewhere (Bhatta & Goel, 1994) that there is a need for more effective and efficient learning methods in design. We represent in design experiences a designer's comprehension of how devices work (i.e., how the structure of a design results in its output behaviors). We represent this comprehension as structure-behavior-function (SBF) models and represent the models of GTMs as behavior-function (BF) models. We hypothesize that the problem-solving context in which learning occurs together with the hierarchical organization of the SBF model of the device helps determine what to generalize from the model. Further, the SBF models lead to a typology of patterns of behavioral regularity over which the generalization process can result in learning GTMs. However, IDEAL does not have a priori knowledge of what the regularities are, but rather, given two design experiences (one without an instance of any generic mechanism and another with the instance of a generic mechanism), it discovers the appropriate regularity by comparing and analyzing the given designs. Also, note that the SBF models define the dimensions and thus constrain the comparison of any two given behaviors (in the form of directed graphs, which can include cycles!). Second, how far a chosen aspect of the device can be generalized. We hypothesize that the similarities in the SBF models of the current design and related designs in a case memory can help determine how far to generalize. Third, what methods can be used for generalization. This is especially relevant when there are multiple, specialized methods for learning different classes of abstractions. We hypothesize that the typology of the patterns of regularity suggested by SBF models can help to determine what strategy to use.

The Learning Task

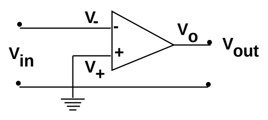
The Problem-Solving Context: IDEAL takes as input a specification of the functional and structural constraints on a desired design, and gives as output a structure that realizes the specified function and satisfies the structural constraints; it also gives an SBF model that explains how the structure realizes that function. A design case in IDEAL specifies (i) the functions delivered by the stored design, (ii) the structure of the design, and (iii) a pointer to the causal behaviors of the design (the SBF model). IDEAL indexes its design cases both by functions that the stored designs deliver and by the structural constraints they satisfy.

IDEAL's **learning task** takes as input a design experience and forms the BF model of a GTM. The input knowledge structure for the learning task is the case-specific SBF model of the given design experience and the output knowledge structure is the case-independent BF model of a GTM. The learned GTM is such that it is an abstraction over certain patterns of behavioral regularity (explained later) observed in the given SBF model and the model of the most similar design in case memory.

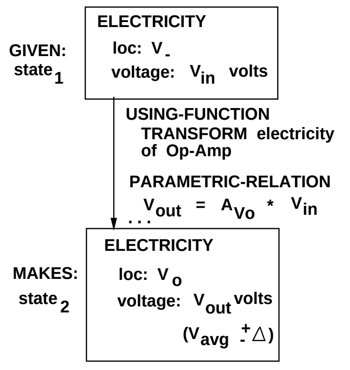
Case-Specific SBF Models

IDEAL's models of specific devices are represented in the form of structure-behavior-function (SBF) models. These models are borrowed from KRITIK system (Goel, 1989) which are based on a *component-substance ontology* (Bylander, 1991) and derived from the functional representation scheme (Sembugamoorthy & Chandrasekaran, 1986; Chandrasekaran, Goel, & Iwasaki, 1993). In this ontology, the **structure** of a device is viewed as constituted of *components* and *substances.* 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, 3 volts,*

[1] IDEAL stands for Integrated "DEsign by Analogy and Learning."



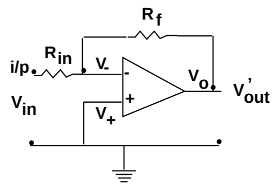
(a) A Basic Op-Amp



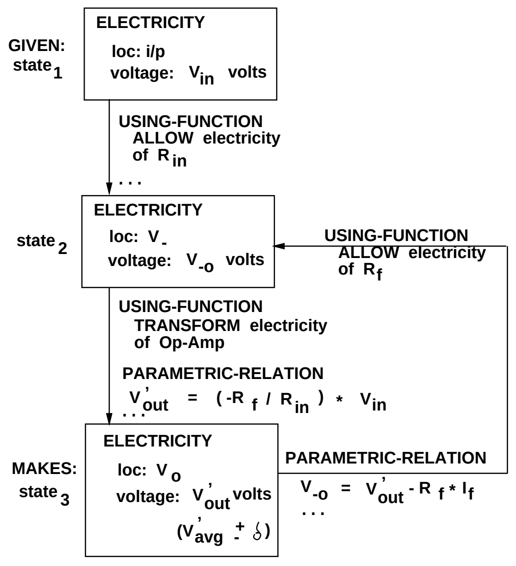
(b) Behavior "Amplify Signal" of the Basic Op-Amp

Figure 1

Note: △ denotes a large fluctuation around an average value, while
δ denotes a small fluctuation.
⊂ denotes set containment.

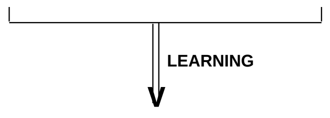


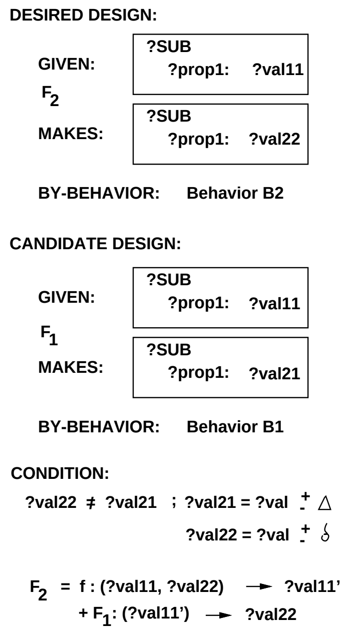
(a) An Inverting Amplifier with Op-Amp



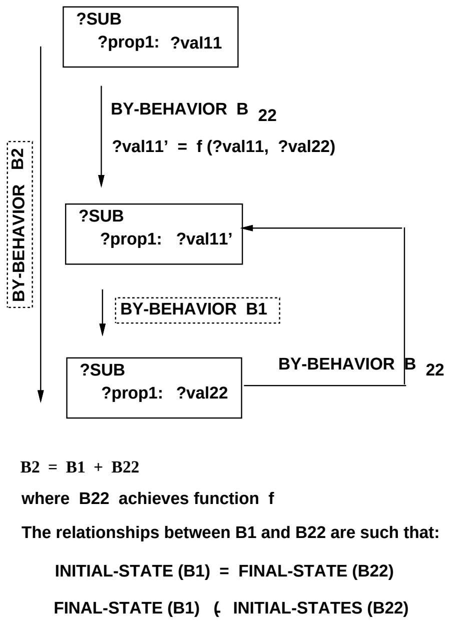
(b) Behavior "Controlled Amplify Signal" of the Inverting Amplifier with Op-Amp

Figure 2





(a) Functional Difference that the Feedback Mechanism reduces


?SUB
?prop1: ?val11
BY-BEHAVIOR B22
?val11' = f (?val11, ?val22)
BY-BEHAVIOR B2
?SUB
?prop1: ?val11'
BY-BEHAVIOR B1
?SUB
?prop1: ?val22
BY-BEHAVIOR B22
B2 = B1 + B22
where B22 achieves function f
The relationships between B1 and B22 are such that:
INITIAL-STATE (B1) = FINAL-STATE (B22)
FINAL-STATE (B1) ⊂ INITIAL-STATES (B22)

(b) Behavior Modification that the Feedback Mechanism suggests

Figure 3

etc. For example, the SBF model of a basic operational amplifier is illustrated in Figure 1 and that of an inverting amplifier in Figure 2. For each device, the structure is shown schematically, its function as the pair of initial and final states of the behavior (indicated by GIVEN and MAKES in the Figures 1 & 2), and the behavior itself as the sequence of states and transitions that explains how the structure achieves the function.

A **function** in SBF models 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. The pair of states indicated by GIVEN and MAKES in Figure 1(b) shows the function "Amplify Signal" of operational amplifier (the function schema is not shown separately due to lack of space). Both the input state and the output state are represented as *substance schemas.* The input state specifies that electricity at `location` V_- in the topography of the device (Figure 1(a)) has the property `voltage` and the corresponding parameter V_{in} `volts.` The output state specifies the parameter of `voltage` is V_{out} `volts` (i.e., $V_{avg} \pm \Delta$ where Δ is a large fluctuation around the average value) at `location` V_o of operational amplifier.

The internal causal **behaviors** of a device are viewed as sequences of *state transitions* between *behavioral states.* 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(b) shows the causal behavior that explains how electricity applied at the input terminal V_- of operational amplifier is amplified at the output terminal V_o. $State_1$ is the preceding state of $transition_{1-2}$ and $state_2$ is its succeeding state. $State_1$ describes the state of electricity at location V_- and so does $state_2$ at location V_o. The annotation USING-FUNCTION in $transition_{1-2}$ indicates that the transition occurs due to the primitive function "**transform** electricity" of operational amplifier. Furthermore, the annotation PARAMETRIC-RELATION indicates the relationships among values of substance properties in different states and the component parameters.

Case-Independent BF Models

Recall that Generic Teleological Mechanisms (GTMs) are one type of abstract (i.e., case-independent) knowledge that designers might use in adaptive design, that is, in modifying an old design by insertion of specific patterns of components (or substructures). GTMs are *teleological* because they result in specific functions and are *generic* because they are case independent. For example, the feedback mechanism takes as input the desired function and the function (where the output value of a substance property is unstable) delivered by an available device, and suggests a structural pattern (i.e., looping back some output to the input and modifying the effective input) of the available device that delivers the desired function.

IDEAL represents its GTMs in the form of BF models. The BF model representation of a GTM encapsulates two types of knowledge: knowledge about the difference between the functions of a known design and a desired design that the GTM can help reduce; and knowledge about modifications to the internal causal behaviors of the known design that are necessary to reduce this difference. For example, Figures 3(a) & 3(b) respectively show these two types of knowledge for a partial model of the feedback mechanism. Figure 3(a) shows the functions F_1 and F_2 respectively of a candidate design available and the desired design, and the conditions underwhich the mechanism is applicable. The model of feedback indicates that the desired behavior (B_2) can be achieved by modifying the candidate behavior (B_1) through setting up the indicated causal relationships between the latter and the additional behaviors (that achieve the subfunctions of F_2 other than F_1 characterized in the applicability conditions of the mechanism). In particular, the feedback mechanism suggests the modification of looping back some output to the input and modifying the effective input to the device. Figure 3(b) shows (both diagrammatically and textually) the relationships in a partial model of the feedback mechanism that IDEAL indeed learns from the two designs of amplifiers. Figure 5(b) in contrast shows the relationships in a more complete model of the feedback mechanism.

The Model-Based Learning Method

Suppose, for instance, IDEAL's case memory has the design (or component) of the basic operational amplifier (op-amp) shown in Figure 1. Note that the output of operational amplifier is dependent on the open loop gain (A_{Vo}, a device parameter) of the operational amplifier and is typically very high (ideally infinite) and unstable. Let us now consider the scenario where IDEAL is presented with a problem of designing an inverting amplifier using an op-amp.[2] This has the function of delivering a specific, controllable output, that is, an output which does not fluctuate much. For instance, the output of the inverting amplifier desired is electricity with a voltage value, $V'_{avg} \pm \delta$, where δ represents a small fluctuation over an average value V'_{avg} (see MAKES state in Figure 2). The fluctuations in the output of a device can in general arise due to several reasons, for instance, due to fluctuations in the input of the device or due to unstable device parameters. In the case of the design of a basic operational amplifier, for example, the fluctuations in the output voltage are due to the device parameter, open-loop gain, A_{Vo} of the op-amp. IDEAL retrieves the design of the basic operational amplifier (Figure 1) because the given functional specification is similar to the amplifying function of the basic op-amp.

Suppose that IDEAL only has a simple strategy such as replacing a component in a past design to de-

[2]An op-amp is always used with feedback, whether it be in inverting or non-inverting configurations (Sedra & Smith, 1991).

liver new functions. In the current scenario, IDEAL would only suggest that the op-amp needs to be replaced with another one that has the desired A_{Vo}, which is not feasible in general! Even if the op-amp can be replaced, doing so will not satisfy the constraint that the output fluctuation be small. Hence, IDEAL fails to modify the retrieved design to generate a design for achieving the new function. It intuitively appears that this task requires an understanding of the mechanism of feedback.

Now the question is whether and how IDEAL can learn a model of the feedback mechanism if it is given the correct design for the current problem. When IDEAL thus fails to solve a problem due to its knowledge conditions, the additional constraint specified (i.e., stable output—no or small fluctuations—being desired), and due to the fact that some components are not available with arbitrary parameters, it has an opportunity to learn. Then, if an oracle presents the correct design that both delivers the desired function and satisfies the additional constraint (the schematic of the structure of the new device is shown in Figure 2(a)) and the SBF model of the new device (shown in Figure 2(b)), IDEAL can form the initial hypothesis for a model of the feedback mechanism. This problem-solving context enables IDEAL to focus on the substructure (op-amp) that amplifies the input voltage for comparing with the corresponding substructure in the old design (basic op-amp). By generalizing over the structural pattern (in this substructure) and the corresponding behavioral segments in both designs and their relationships with the others in the new design, it learns the feedback mechanism. We will now focus on the learning of the feedback mechanism.

The learning method is model-based in that the SBF models of the design cases provide the content for generalizing over the patterns of regularity in the device structure and device behavior. The representation vocabulary of the SBF models further defines the dimensions along which two behaviors can be compared and leads to several classes of regularity based on the "cause-effect" relationships between behavior segments.[3] For instance, behavioral states, transitions, and behavioral segments (state-transition-state) are some dimensions at a top level along which two behaviors can be compared. Within comparing two behaviors along these dimensions a few lower level dimensions are substances and components, their properties and values, and primitive functions. A few patterns of regularity in device behaviors B_1 and B_2 are illustrated in Figure 4; B_1 is the behavior of an available or candidate design and B_2 is the behavior of the new or desired design. In the following discussion, F_1 denotes the function of the candidate design and F_2 that of the desired design. For the following general description of IDEAL's learning method, the reader is referred to the complete model of the feedback mechanism shown in Figure 5 unless otherwise mentioned.

The learning method first traverses the two fo-

[3]A behavior segment is a partial sequence of states and transitions in a behavior. The smallest behavior segment will have just two states and a transition between them.

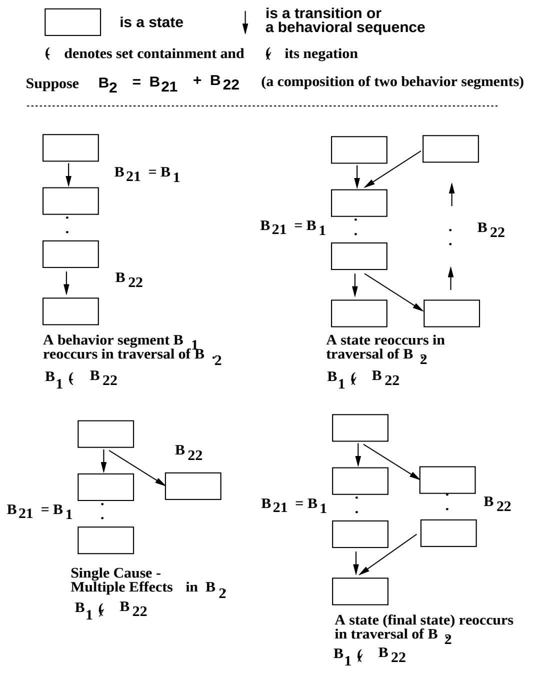

is a state
is a transition or a behavioral sequence
∈ denotes set containment and ∉ its negation
Suppose $B_2 = B_{21} + B_{22}$ (a composition of two behavior segments)
$B_{21} = B_1$
B_{22}
A behavior segment B_1 reoccurs in traversal of B_2
$B_1 \in B_{22}$
$B_{21} = B_1$
B_{22}
A state reoccurs in traversal of B_2
$B_1 \notin B_{22}$
B_{22}
$B_{21} = B_1$
Single Cause - Multiple Effects in B_2
$B_1 \notin B_{22}$
$B_{21} = B_1$
B_{22}
A state (final state) reoccurs in traversal of B_2
$B_1 \notin B_{22}$

Figure 4: **A Few Patterns of Regularity in Device Behaviors**

cused behaviors and compares them for similarity. When the behavior of the old design (B_1) matches with (or is similar to) some segment in the new device behavior (B_2), then there is an opportunity for IDEAL to learn a generic mechanism that specifies how to modify a behavior like B_1 to get a behavior like B_2 that achieves the function like F_2. Suppose that $B_2 = B_{21} + B_{22}$, a composition of two behavior segments, and that B_1 matches with B_{21}. Under such conditions, IDEAL can form a generic mechanism only on the basis of its analysis of differences between B_1 and B_2, in particular, the relationships between B_1 and B_{22}. IDEAL's hypothesis is that the difference in the functions of the two devices ($F_2 - F_1$) can be attributed to the difference in the behaviors ($B_2 - B_1$, which is the additional behavior B_{22}) and the relationships between B_1 and B_{22}. While traversing the two focused behaviors, IDEAL can recognize the end of a behavior, recurrence of states, single cause-multiple effects (i.e., a fork in the directed graph), and multiple causes-single effect (i.e., a join in the directed graph), and repetition of a behavior segment, without an explicit a priori knowledge of them. From these basic patterns, IDEAL discovers the regularities in the relationships between B_1 and B_{22} and forms mechanisms such as cascading, feedback, and feedforward. IDEAL then generalizes over the specifics of these relationships so that the learned mechanism is useful in different problem-solving contexts.

In addition, in order for a new mechanism to be useful, IDEAL needs to identify the applicability conditions for the mechanism. Because IDEAL believes that the relationships between B_1 and B_{22} to be responsible for the difference in the candidate and desired functions, it forms the decomposability condition on the desired function as one of the applicability conditions. For instance, when a desired function, F_2, is specified, and a candidate design delivers F_1, one applicability condition for using "a" generic mechanism is to check if F_2 can be decomposed into F_1 and any other subfunctions and precisely what those additional subfunctions are. Hence it finds what the subfunctions besides the candidate function (F_1) that the desired function can be decomposed into are (i.e., F_2 = f + F_1 + g or F_2 = f + F_1 or F_2 = F_1 + g). Since in general segments in B_{22} can be distributed partly preceding B_{21} and partly succeeding B_{21}, there can only be at most two subfunctions other than F_1 (or multiples of it)—one subfunction f that is an abstraction over the behavior segments that precede B_{21} (i.e., B_{22-1}) and the other subfunction g that is an abstraction over the behavior segments that succeed B_{21} (i.e., B_{22-2}). By tracing B_{22} back from the initial state of B_{21} and tracing it forward from the final state of B_{21} in B_2, IDEAL can identify the segments B_{22-1} and B_{22-2} and find their functional abstractions f and g in order to learn an applicability condition for the new mechanism. IDEAL analyzes in this manner because it would enable discrimination among competing mechanisms in terms of these subfunctions of a desired function, while using the mechanisms in later problem solving. For instance, f and g would be different for feedback and feedforward mechanisms. Thus IDEAL describes all the generic mechanisms it learns in a uniform representation, that is, in terms of relationships between candidate and desired functions and the corresponding behaviors. In sum, a generic mechanism specifies how to compose the candidate behavior (that achieves F_1) with the behaviors that achieve the subfunctions f and g to generate a behavior that achieves the desired function F_2 where F_2 can be decomposed into f, F_1, and g.

In the current problem-solving scenario, applying the above learning method, IDEAL finds that the behavior segment ($state_2 \rightarrow state_3$ in Figure 2(b)) in B_2, the behavior of the inverting amplifier, matches with B_1, the behavior of the basic op-amp ($state_1 \rightarrow state_2$ in Figure 1(b)). Continuing to traverse the additional segments in B_2, it discovers that there is a cycle (or loop) in B_2 and picks out the relationships between the matching segment B_{21} (i.e., $state_2 \rightarrow state_3$) and the preceding behavior segment (i.e., $(state_1, state_3) \rightarrow state_2$ which constitutes B_{22-1}) in B_2. The functional abstraction over this preceding segment is f and it becomes part of the decomposition of F_2. Since in the specific behavior of the new design there are no additional states succeeding the final state of the matching segment (i.e., after $state_3$ in $state_2 \rightarrow state_3$) that are not already taken into account in B_{22-1}, B_{22-2} in this learning situation is *null* and hence there is no subfunction g in the decomposition of F_2. IDEAL then generalizes over the specific substances, properties and values, and the relationships to form an initial hypothesis for a generic mechanism, which is the mechanism of feedback. The model of the learned (more precisely, *hypothesized*) feedback mechanism

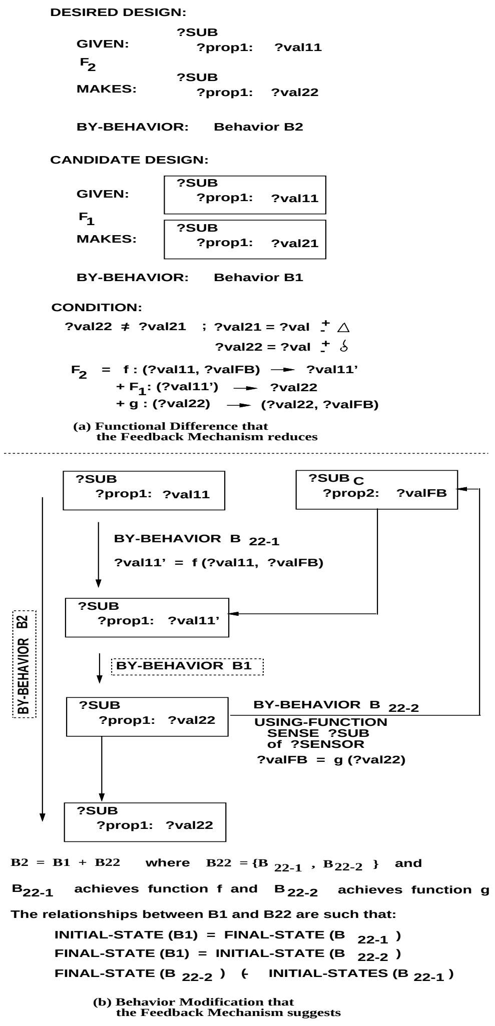

DESIRED DESIGN:
GIVEN:
?SUB
?prop1: ?val11
F2
MAKES:
?SUB
?prop1: ?val22
BY-BEHAVIOR: Behavior B2
CANDIDATE DESIGN:
GIVEN:
?SUB
?prop1: ?val11
F1
MAKES:
?SUB
?prop1: ?val21
BY-BEHAVIOR: Behavior B1
CONDITION:
?val22 ≠ ?val21 ; ?val21 = ?val ± △
?val22 = ?val ± δ
F2 = f : (?val11, ?valFB) → ?val11'
+ F1: (?val11') → ?val22
+ g : (?val22) → (?val22, ?valFB)
(a) Functional Difference that the Feedback Mechanism reduces
?SUB ?prop1: ?val11
?SUB C ?prop2: ?valFB
BY-BEHAVIOR B 22-1
?val11' = f (?val11, ?valFB)
?SUB ?prop1: ?val11'
BY-BEHAVIOR B1
BY-BEHAVIOR B2
?SUB ?prop1: ?val22
BY-BEHAVIOR B 22-2
USING-FUNCTION SENSE ?SUB of ?SENSOR
?valFB = g (?val22)
?SUB ?prop1: ?val22
B2 = B1 + B22 where B22 = {B 22-1 , B22-2 } and
B22-1 achieves function f and B 22-2 achieves function g
The relationships between B1 and B22 are such that:
INITIAL-STATE (B1) = FINAL-STATE (B 22-1)
FINAL-STATE (B1) = INITIAL-STATE (B 22-2)
FINAL-STATE (B 22-2) ⊂ INITIAL-STATES (B 22-1)
(b) Behavior Modification that the Feedback Mechanism suggests

Figure 5: **A Complete BF Model of the Feedback Mechanism**

and its index are shown in Figure 3.

Note that the feedback mechanism IDEAL learned in the current scenario (Figure 3) is only a partial model of the feedback mechanism because it assumes that the controlling or feedback substance is same as the controlled or output substance (?Sub)), which is not true in general.[4] A more complete model of the feedback mechanism as illustrated in Figure 5 ought to distinguish between the feedback substance ($?Sub_C$) and the controlled substance ($?Sub$) as well as consider the more general decomposition of F_2 in terms of f, F_1, and g. When the feedback substance and the controlled substance are different, the decomposition of F_2 will have the subfunction g and the subfunction g would involve sensing the fluctuations in a property value of the controlled substance.

Note also that the feedback mechanism IDEAL

[4]In fact, IDEAL does not even recognize that the feedback and controlled substances could be different because the current design experiences do not indicate that.

had learned does not capture the subtleties of open loop feedback and closed loop feedback. Even Figure 5 shows only a model of closed loop feedback. In order to learn those distinctions, however, IDEAL requires more design experiences in which the substances fed back are different and the points in the device topology to where they are fed back are different. Thus acquiring a complete model of the feedback mechanism (or, in other words, all the different types of feedback mechanism) may involve solving a number of design problems incrementally and revising the hypothesized mechanism.

Conclusions

We have presented a computational model of how generic mechanisms can be learned from design experiences incrementally by generalization and demonstrated it in the context of the design of physical devices. Case-specific SBF models of devices in the design experiences provide the content for learning the BF models of generic mechanisms. The internal organization of the SBF models (e.g., functional, structural, and behavioral decomposition) together with the problem-solving context provides the constraints for learning by generalization. The vocabulary of the SBF models also provides the dimensions along which two devices can be compared and thus constrain the comparison. Further, similarities between regularities in experiences determine how abstract a learned generic mechanism can be.

References

Bhatta, S. & Goel, A. 1993. Learning Generic Mechanisms from Experiences for Analogical Reasoning. In *Proc. of the Fifteenth Annual Conf. of the Cog. Sci. Soc.*, 237–242.

Bhatta, S. & Goel, A. 1994. Discovery of Physical Principles from Design Experiences. *AI EDAM*, **8**(2):113-123.

Bylander, T. 1991. A Theory of Consolidation for Reasoning about Devices. *Intl. Jnl. of Man-Machine Studies* **35**(4):467-489.

Catrambone, R. & Holyoak, K. 1989. Overcoming Contextual Limitations on Problem-Solving Transfer. *Jnl. of Experimental Psy.: Learning, Memory, and Cognition* **15**(6):1147-1156.

Chandrasekaran, B., Goel, A., & Iwasaki, Y. 1993. Functional Representation As Design Rationale. *IEEE Computer,* January:48-56.

Falkenhainer, B. 1989. Learning from Physical Analogies: A Study in Analogy and the Explanation Process. Ph.D. diss., Department of Comp. Sci., University of Illinois, Urbana.

Gick, M.L. & Holyoak, K.J. 1983. Schema Induction and Analogical Transfer. *Cognitive Psychology* **15**:1-38.

Goel, A. 1989. Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving. Ph.D. diss., Dept. of Comp. and Info. Sci., The Ohio State University.

Goel, A. 1991. A Model-Based Approach to Case Adaptation. In *Proc. of the Thirteenth Annual Conf. of the Cog. Sci. Soc.,* 143-148.

Greiner, R. 1988. Learning by Understanding Analogies. *Artificial Intelligence,* **35**:81-125.

Maher, M. & Zhao, F. 1987. Using Experiences to Plan the Synthesis of New Designs. In J. Gero, (Ed.), *Expert Systems in Computer-Aided Design*, 349–369. North-Holland, Amsterdam, Netherlands.

Sedra, A. & Smith, K. 1991. *Microelectronic Circuits.* Ch. 3, 86–149. Holt, Rinehart and Winston, Inc., New York.

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

Stroulia, E. & Goel, A. 1992. Generic Teleological Mechanisms and their Use in Case Adaptation. In *Proc. of the Fourteenth Annual Conf. of the Cog. Sci. Soc.,* 319-324.