Learning Generic Mechanisms from Experiences for Analogical Reasoning*

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

Abstract

Humans appear to often solve problems in a new domain by transferring their expertise from a more familiar domain. However, making such 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 used by designers. However, one important yet unexplored issue is where these generic mechanisms come from. We hypothesize that they are acquired incrementally from problem-solving 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 show that mental models 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 show how the model-based learning method integrated with similarity-based learning addresses the issues in generalization from experiences.

Introduction

Analogy is often believed to play an important role in reasoning underlying innovation and creativity. Analogies can be of different types: within-problem, cross-problem but within-domain, and cross-domain. We are interested in studying cross-domain analogies. Psychological research shows that humans use abstractions in making cross-domain analogies (e.g., Gick & Holyoak, 1983; Catrambone & Holyoak, 1989). Some of the issues of interest then are how reasoning is mediated by the abstractions (shared between the source and target domains) and how those abstractions are learned. We explore the latter issue in the context of the design of physical devices such as electric circuits and heat exchangers. Our goal is to build a computational model that can account for these phenomena and use it to generate testable predictions about designers' behavior.

Goel (1989) has proposed models of generic teleological mechanisms (GTMs), such as cascading, feedback, and feedforward, as one type of abstract knowledge that designers use in case-based design. 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. Stroulia and Goel (1992) have shown that GTMs indeed are useful in non-routine adaptive design. But one important yet unexplored issue is how these GTMs are acquired. Our hypothesis is that they are acquired incrementally from problem-solving experiences in familiar domains by generalization over patterns of regularity. For instance, a designer may acquire from examples in the domain of electric circuits a model of cascading, and when and how to cascade a number of similar components together (i.e., to connect multiple components to amplify the overall delivered function). The designer can then use that model for designing in a different domain such as the domain of heat exchangers.

Generalization from experiences raises three important issues. First is the issue of relevance, that is, the issue of deciding what to generalize from an experience. 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 propose that the problem-solving context in which learning occurs together with the hierarchical organization of the SBF model of the device help determine what to generalize from the model. Further, the SBF models lead to a typology of behavioral patterns over which the generalization process can result in learning GTMs. Second, how far a chosen aspect of the device can be generalized. We show that the similarities in the SBF models of the cur-

*This work has been supported by research grants from ONR (contract N00014-92-J-1234), NSF (grant C36-688), Northern Telecom, Georgia Tech Research Corporation, and a CER grant from NSF (grant CCR-86-19886), and equipment grants and donations from IBM, Symbolics, and NCR.

rent design and related designs in a case memory can help determine how far to generalize. Third, what methods can be used for generalization. We show that a typology of the patterns of regularity in SBF models can help to determine what strategy to use.

The system IDEAL[1] implements the proposed learning method. We evaluate the learning method by showing how the GTMs learned in one domain can facilitate designing in another domain.

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 regularity (explained later) observed in the structure and behavior of the given SBF model and the model of the most similar experience 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 based on a *component-substance ontology* (Bylander, 1991). 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*, etc. This ontology gives rise to a *behavioral representation language* (Goel, 1989) for describing the SBF model of a design that is a generalization on functional representation scheme (Sembugamoorthy & Chandrasekaran, 1986; Chandrasekaran, Goel, & Iwasaki, 1993). The constituents of the SBF model are described below.

Structure: The structure of a design is expressed in terms of its constituent components and substances and the interactions between

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

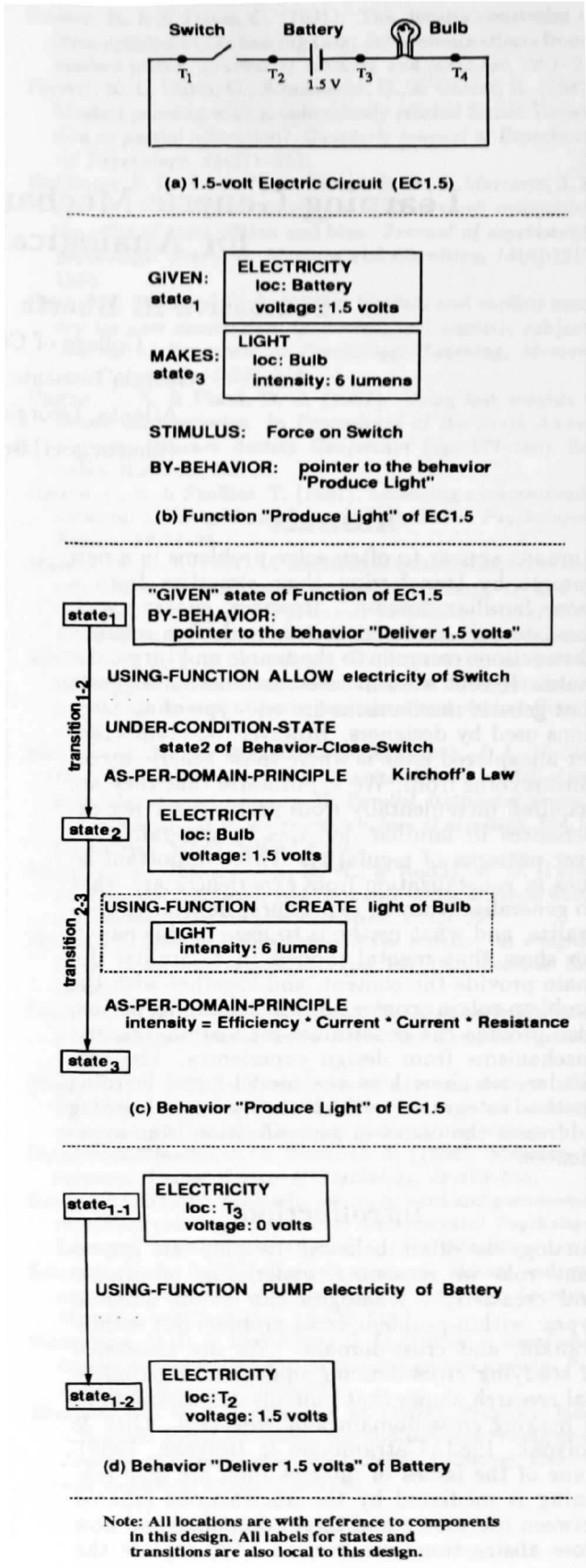

Figure 1: **Design of A 1.5-volt Electric Circuit (EC1.5)**

them. Figure 1(a) shows the structure of a `1.5-volt electric circuit` (EC1.5) schematically.

Function: A function 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(b) shows the function "Produce Light" of EC1.5. Both the input state and the output state 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.

Behavior: 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(c) shows the causal behavior that explains how electricity in Battery is transformed into light in Bulb. $State_2$ is the preceding state of $transition_{2-3}$ and $state_3$ is its succeeding state. $State_1$ 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 annotation 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 state transitions at a different level as shown in Figure 1(d).

Case-Independent BF Models

Generic Teleological Mechanisms (GTMs) are one type of knowledge that designers use in adaptive design, that is, in modifying an old design by insertion of specific patterns of components (or substructures) (Stroulia & Goel, 1992). Examples of GTMs are cascading, feedback, and feedforward. GTMs are *teleological* because they result in specific functions and are *generic* because they are case independent. For example, the cascading mechanism takes as input the desired function and the function (with a lesser range) delivered by an available device, and suggests a structural pattern (i.e., the replication) of the available device that delivers the desired function. Further, the cascading mechanism can be instantiated in any specific device that satisfies its applicability conditions. For instance, one applicability condition is that the functions delivered by each replicated device should add up to give the desired function (i.e., the replication should be *functionally additive*). More precisely, the condition is that the smaller parametric transformation delivered by each replicated device should sum up to provide the desired larger transformation.

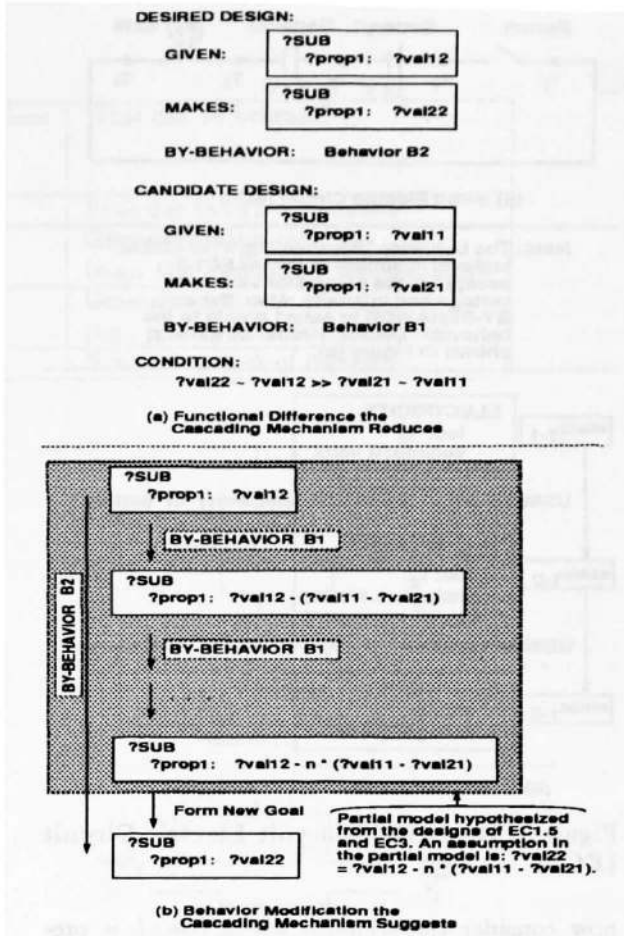

Figure 2: **BF Model of the Cascading Mechanism**

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 2(a) & 2(b) respectively show these two types of knowledge for the cascading mechanism. The model of cascading indicates that a behavior can be replicated as much as possible to achieve a desired function and finally a goal be formed to find a component that can deliver the residual function. This additional component is needed when the desired function is not an integral multiple of the function of each replicated device.

The Learning Method

Suppose, for instance, IDEAL's case memory has the design of EC1.5 shown in Figure 1. Let us

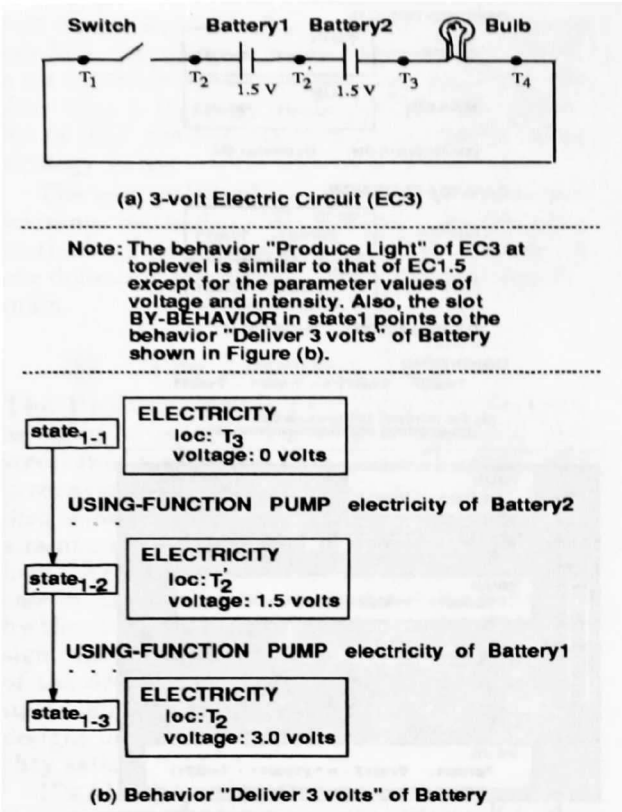

Figure 3: **Design of A 3-volt Electric Circuit (EC3)**

now consider the scenario where IDEAL is presented with a problem of designing a 3-volt electric circuit (EC3) that delivers the function "produce light of intensity 12 lumens in the bulb when the switch is closed, given that there is electricity with a voltage of 3 volts in the battery" and satisfies the structural constraint "the design cannot have a single 3-volt battery." IDEAL retrieves the design case EC1.5 because the given functional specification is similar to the function of EC1.5. However, IDEAL may know only how to replace a component in a past design to solve the current problem. The component-replacement plan specifies how to replace the component that is responsible for the functional difference by a new component that reduces the functional difference and thus enables the overall device to deliver the desired function. In such cases, IDEAL fails to solve the current problem due to the structural constraint specified. Then, if an oracle presents the correct solution that both delivers the desired function and satisfies the structural constraint (the schematic of the structure of the new device is shown in Figure 3(a)), IDEAL learns how the new device behaves (a segment is shown in Figure 3(b)) by revising the behavior of EC1.5. This problem-solving context enables IDEAL to focus on the substructure that delivers the required voltage for comparing with the corresponding substructure in the old case EC1.5. By generalizing over the structural pattern (in this substructure) and the corresponding behavioral segments, it learns the cascading mechanism. We will now focus on the learning of the cascading 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 leads to several classes of regularity, a few of which that are relevant to learning cascading mechanism are: (i) repetition of behavioral segments, that is, a sequence of state-transitions repeats several (say, n) times in the overall device behavior; since a behavior typically corresponds to a structural part (i.e., a component), the corresponding structural regularity is the repetition of the structural part; (ii) repetition of a range of input-output transformation, that is, the same amount of parameter transformation repeats several (say, n) times in the device behavior. The two variables of interest for generalization then are the *range of transformation* (r) and *number of repetitions of same structure* (n). Given the task of learning from two design cases and that there are two variables, four different situations are possible as shown in Table 1. In this paper we will focus on situation **2** only.

The learning method first traverses each focused behavior in the given two designs to notice the above types of regularities, in particular, to identify the values for n and r. Then it compares the values for the two variables in both the designs and generalizes over them if any similarity exists. The first step of the learning method can be facilitated by indexing from the component into the behavioral segments in which some function of the component plays a role.

In the above problem-solving scenario, the problem-solving context indicates that the behavioral segments to focus on for learning are those that correspond to the function of **Battery** in the two designs, EC1.5 and EC3. Applying the above learning method, it is easy to identify that the learning situation here is **2** shown in Table 1. Generalizing over the number of repetitions and variablizing the range of parameter transformation, IDEAL hypothesizes a GTM that would help in a problem-solving context similar to the current one. The model of the learned (more precisely, *hypothesized*) cascading mechanism and its index are shown in Figure 2 (representations in (a) and the shaded region of (b)); the functional difference that the cascading mechanism reduces is the index for the mechanism.[2]

IDEAL can revise the hypothesized model into a more complete one when it solves a new design problem whose solution has a *structural pattern* that is an instance of the complete cascading mechanism. Thus acquiring a complete model of the cascading mechanism may involve solving a number of design problems incrementally.

[2] A new piece of knowledge learned is futile unless its applicability conditions (or *indices* as we call them) are also learned.

Table 1: **Situations of Regularity Between Similar Components in Two Designs**

Situation	Range of Input-Output Transformation in both designs, r	Number of Repetitions in both designs, n	What can be Learned?
1.	equal	equal	None due to lack of variation.
2.	**equal**	**not equal**	**Generalization over n. (e.g., the cascading mechanism)**
3.	not equal	equal	Generalization over r. (e.g., prototypical device models)
4.	not equal	not equal	None due to lack of regularity.

Evaluation

One method for evaluating the learning is to show how the learned mechanisms can affect IDEAL's performance task of designing physical devices. In particular, does it enable IDEAL to transfer design knowledge from one domain (say, electric circuits) to another domain (say, heat exchangers)?

We have tested IDEAL with several designs from the domain of electric circuits and heat exchangers. In one experiment, we gave IDEAL designs of electric circuits such as those illustrated in this paper. IDEAL learned the mechanism of cascading, indexed it by the applicability conditions of the mechanism, and stored it in its memory. Then we gave IDEAL a design problem in the domain of heat exchangers. This problem, relative to IDEAL's knowledge, was such that in order to solve it IDEAL would need to evoke the cascading mechanism. We observed that IDEAL noticed the difference between the desired function and the function of an available device. It then used the functional difference as a probe into its memory, retrieved the cascading mechanism, and solved the new problem by instantiating the retrieved mechanism. More specifically, Figure 4 illustrates how IDEAL instantiated the cascading mechanism learned from the two designs, EC1.5 and EC3, in the water pumps in designing a nitric acid cooler that provides a higher range of cooling (i.e., T_1-T_2'). (Stroulia & Goel, 1992) provides more details of the adaptation process.

Together, these experiments indicate the utility and effectiveness of our model-based method for learning GTMs: the SBF models enable learning of GTMs in one domain and the learned BF models of GTMs facilitate designing in another. We are currently testing IDEAL with design problems from other domains such as reaction wheel assemblies, and for other mechanisms such as feedback and feedforward.

Related Work

This work on IDEAL evolves from our earlier work on KRITIK (Goel, 1989). IDEAL's component-substance ontology, SBF models, and behavioral representation language all are borrowed from KRITIK. The problem-solving component of

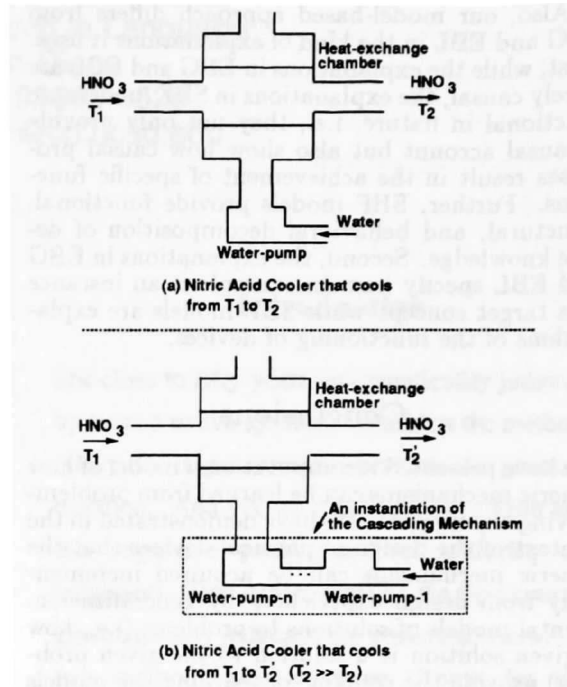

Figure 4: **Designs of Nitric Acid Cooler**

IDEAL evolves from KRITIK2 (Stroulia *et al.*, 1992).

Learning Task: Few computational models of analogical reasoning have addressed learning of high-level abstractions. Birnbaum and Collins (1988) discuss the need for acquisition of abstract strategies that enable transfer of expertise from one domain to another. Their work uses explanation-based learning (EBL) techniques in failure-driven learning of abstract strategies for game playing (e.g., chess). GTMs in our work are similar to their abstract strategies in that GTMs also act as abstract plans for solving design-adaptation problems. However, Birnbaum and Collins view the abstract strategies to be useful only in accessing a relevant experience, that is, they view cases to be indexed by these abstract concepts. In contrast, in our theory, abstract models are useful in both the access and transfer stages of analogical reasoning. Moreover, in

our approach learning is not only failure-driven but it also occurs from successful experiences.

Learning Method: Our model-based approach to learning is similar to Winston's model (1982) which shows that learning can be done by analogically transferring causal links in the explanation of an example to the target "concept." Our approach is also similar to explanation-based methods such as EBG (Mitchell, Keller, & Kedar-Cabelli, 1986) and EBL (DeJong & Mooney, 1986) in using explanations (SBF models) to constrain the learning of concepts. However, most of these systems assume some knowledge of the target concept *a priori;* our model-based approach attempts to "discover" them.

Also, our model-based approach differs from EBG and EBL in the kind of explanations it uses. First, while the explanations in EBG and EBL are purely causal, the explanations in SBF models are functional in nature, i.e., they not only provide a causal account but also show how causal processes result in the achievement of specific functions. Further, SBF models provide functional, structural, and behavioral decomposition of device knowledge. Second, the explanations in EBG and EBL specify how an example is an instance of a target concept while SBF models are explanations of the functioning of devices.

Conclusions

We have presented a computational model of how generic mechanisms can be learned from problem-solving experiences. We have demonstrated in the context of the design of physical devices that the generic mechanisms can be acquired incrementally from design experiences by generalization. Mental models of solutions to problems (i.e., how a given solution is *a* solution to the given problem) provide the content for learning the models of generic mechanisms. The internal organization of mental models (e.g., functional, structural, and behavioral decomposition) together with the problem-solving context provides the constraints for learning by generalization. Further, similarities between regularities in experiences determine how abstract a learned generic mechanism can be.

Elsewhere we show how our computational model also accounts for the acquisition of other types of "abstract concepts," such as mental models of physical principles, physical processes, and device prototypes (Bhatta & Goel, 1992).

Finally, from the computational model we can predict that if they have the models of specific devices, human designers can easily learn the models of generic mechanisms from their design experiences and use the learned mechanisms for making cross-domain analogies.

Acknowledgments. We thank Eleni Stroulia for her contributions to KRITIK2 and Mimi Recker for her comments on an earlier draft of this paper.

References

Bhatta, S. & Goel, A. 1992. Discovery of Physical Principles from Design Experiences. In J.M. Zytkow (Ed.), *Proceedings of the ML-92 Workshop on Machine Discovery,* 77-81. Aberdeen, Scotland, UK.

Birnbaum, L. & Collins, G. 1988. The Transfer of Experience Across Planning Domains Through the Acquisition of Abstract Strategies. In J. Kolodner (Ed.), *Proceedings of the DARPA Workshop on Case-Based Reasoning,* 61-79. Clearwater Beach, FL.

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

Catrambone, R. & Holyoak, K. 1989. Overcoming Contextual Limitations on Problem-Solving Transfer. *Journal of Experimental Psychology: 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.

DeJong, G. & Mooney, R. 1986. Explanation-Based Learning: An Alternative View. *Machine Learning* 1(2):145-176.

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 Computer and Information Science, The Ohio State University.

Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. 1986. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1(1):47-80.

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 *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society,* 319-324. Bloomington, IN.

Stroulia, E., Shankar, M., Goel, A., & Penberthy, L. 1992. A Model-Based Approach to Blame-Assignment in Design. In J.S. Gero (Ed.), *Proceedings of the Second International Conference on AI in Design,* 519-537. Pittsburgh, PA.

Winston, P.H. 1982. Learning New Principles from Precedents and Exercises. *Artificial Intelligence* 19(3):321-350.