Available online at www.sciencedirect.com
SCIENCE DIRECT®
Advanced Engineering Informatics 18 (2004) 85–94
ADVANCED ENGINEERING INFORMATICS
www.elsevier.com/locate/aei

Use of design patterns in analogy-based design

Ashok K. Goel[a,1], Sambasiva R. Bhatta[b,*]

[a]College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, USA
[b]Verizon Communications, 500 Westchester Avenue, White Plains, NY 10604, USA

Abstract

Design patterns specify generic relations among abstract design elements. We hypothesize that design patterns are productive units of analogical transfer in design. We describe a normative theory of analogy-based design called model-based analogy (or MBA) that transfers design patterns from source cases to target problems. In particular, for the domain of physical devices, we identify a class of design patterns, called generic teleological mechanisms (or GTMs), that specify generic functional relations in, and abstract causal structure of, a class of devices. While GTMs provide a partial content account of analogical transfer, MBA provides a process account of acquisition, access, and use of GTMs. In particular, MBA shows how structure–behavior–function (SBF) models of specific designs enable the acquisition of GTMs, which are represented as behavior–function (BF) models, and how goals of adapting familiar designs to meet new design requirements result in the access, transfer, and use of previously learned GTMs. We describe how the IDeAL system instantiates and evaluates the MBA theory.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Design; Analogy; Ontology; Design patterns; Functional models

1. Background, motivations and goals

\Design patterns are oft-studied entities in the design literature. Alexander [1] for example, analyzed evolution of designs of village centers in rural India in terms of design patterns. We characterize design patterns as specifying generic relations among abstract design elements. The relations are generic in that they are independent of any specific design situation and the elements are abstract in that they do not refer to any specific physical structure. Let us focus on the domain of physical devices, i.e. the domain of teleological artifacts instantiatable in physical form. In this domain, we characterize design patterns as specifying the generic functional relations in, and the abstract causal structure of, a class of devices. The abstract concept of feedback in control systems provides one example of such a functional and causal pattern; a generic mechanism for transforming translational motion into rotational motion is another example. The design pattern of feedback specifies both the generic function it achieves (e.g. regulation of a device output, given possible fluctuations in the device input) independent of any specific design situation, and the abstract causal structure that achieves it (e.g. transmission of information about fluctuations in the device output to a device control input) without reference to the physical structure of any particular device. We call these functional and causal design patterns *generic teleological mechanisms* (or GTMs).

We posit that design patterns are productive units of analogical transfer in design. That is, design patterns are what gets learned from one design situation and transferred to another situation. For the domain of device design, we hypothesize that GTMs are productive units of analogical transfer in the domain of device design. We have developed a normative theory of conceptual device design called *model-based analogy* (or MBA) based on this notion. While GTMs provide a content account of analogical transfer, MBA provides a process account that specifies how the transfer occurs, i.e. how GTMs are acquired, accessed, and used in analogical design. In particular, it shows how structure–behavior–function (SBF) models of specific designs enable the acquisition of GTMs, and how goals of adapting familiar designs to meet new design requirements result in the access and use of previously learned GTMs. We have developed an

* Corresponding author. Tel.: +1 914 644 5058.
E-mail addresses: goel@cc.gatech.edu (A.K. Goel), bhatta@basit.com (S.R. Bhatta).
[1] Tel.: +1 404 894 4994.

1474-0346/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2004.09.003

ontology of structure, behavior, and function of physical devices. This ontology provides the vocabulary for expressing SBF models. GTMs are expressed as behavior–function (BF) models using a subset of the SBF vocabulary.

We have instantiated and evaluated the MBA theory in an operational computer program called IDeAL. In this article, we briefly describe the MBA theory, focusing on the hypothesis about GTMs forming a productive unit of analogical transfer in device design. The ontology of the structure, behavior, and function of physical devices is described in detail in [13]. The acquisition of GTMs represented as BF models from SBF device models is described in [3]. This paper focuses specifically on the transfer of GTMs from a source design analog to a target design problem.

2. Model-based analogy

The computational process of MBA takes as input a specification of a target design problem in the form of the functional requirements and structural constraints on a desired design, and gives as output a solution in the form of a structure that realizes the specified function(s) and also satisfies the structural constraints. In addition, MBA gives an SBF model that explains how the structure realizes the desired function.[2]

Fig. 1 illustrates a part of the MBA process that pertains to GTMs. In this figure, the boxes represent the subtasks of analogical design such as the retrieval of source analogue and analogical transfer; the arrows between the boxes represent the process flow between the subtasks; the ovals represent the knowledge in memory such as design analogues; and the dotted ovals represent methods to achieve the subtasks. A stored design analogue in this process specifies (i) the functions delivered by the known design, (ii) the structure of the design, and (iii) a pointer to the causal behaviors of the design (the SBF model). The design analogues are indexed both by the functions that the stored designs deliver and by the structural constraints they satisfy.

MBA considers a source analogue to be an exact match for a target problem if both the input states and output states of the functions in the analogue and the problem are identical along with the property values and their constraints. If and when IDeAL is unable to find a source analogue that exactly matches the target problem, it spawns reasoning goals for adapting the source design. Different types of functional differences between the target and the source lead to different types of adaptation goals, some requiring only simple modifications (such as parameter tweaks) and some others requiring more complex modifications (such as topological changes). In order to control the reasoning involved in making complex modifications, MBA

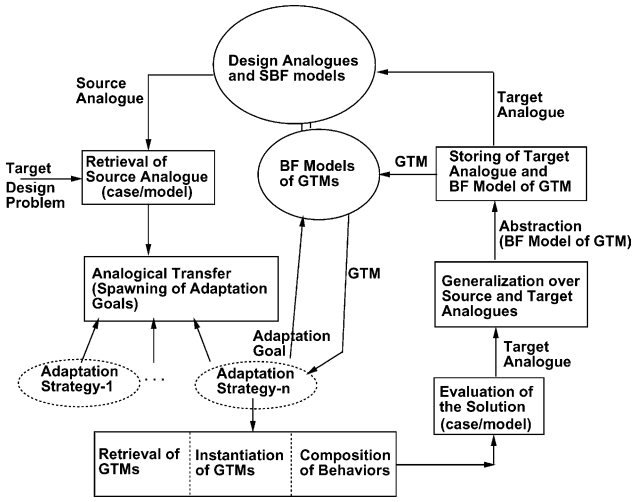

Fig. 1. IDeAL's process of analogical design using GTMs.

requires knowledge that can encapsulate the relationships between candidate modifications and their causal effects. In device design, design patterns, and, in particular, GTMs, provide such knowledge. Therefore, MBA uses the knowledge of GTMs in modifying device topology in the source design. IDeAL evaluates a candidate design by qualitative simulation of the model it generates by instantiating an applicable GTM.

2.1. SBF models of devices

IDeAL represents its comprehension of specific design cases (i.e. device models) in a structure–behavior–function (SBF) language [12]. This language provides conceptual primitives for representing and organizing knowledge of the structures, behaviors, and functions of a device. In this representation, 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, 3 V, etc. Fig. 2(a) and (b) illustrates the SBF model of a simple amplifier and Fig. 2(c) and (d) that of an inverting amplifier. For each device, the structure, its function, and the behavior that achieves the function are shown. For exposition, the structure here is illustrated diagrammatically. IDeAL, however, internally represents the structure using symbols, not diagrams. The structure of a device in SBF models is represented hierarchically in terms of its constituent structural elements and relations among them such as *part-of*, *includes*, and *parallelly-connected*. For instance, the structure of the simple amplifier (shown in Fig. 2(a)) is represented symbolically as having components *resistor* (R_{in}) and *operational amplifier* (*Op-Amp*), and the structural relation *serially-connected* between them at the location V_-.

A **function** in the SBF models is a behavioral abstraction and is represented as a schema that specifies the behavioral

[2] Note that although some figures in this paper contain diagrams of devices for illustration, MBA does not reason diagrammatically. Instead, the input to the system is in the form of schemas or 'frames' representing functional requirements and structural constraints.

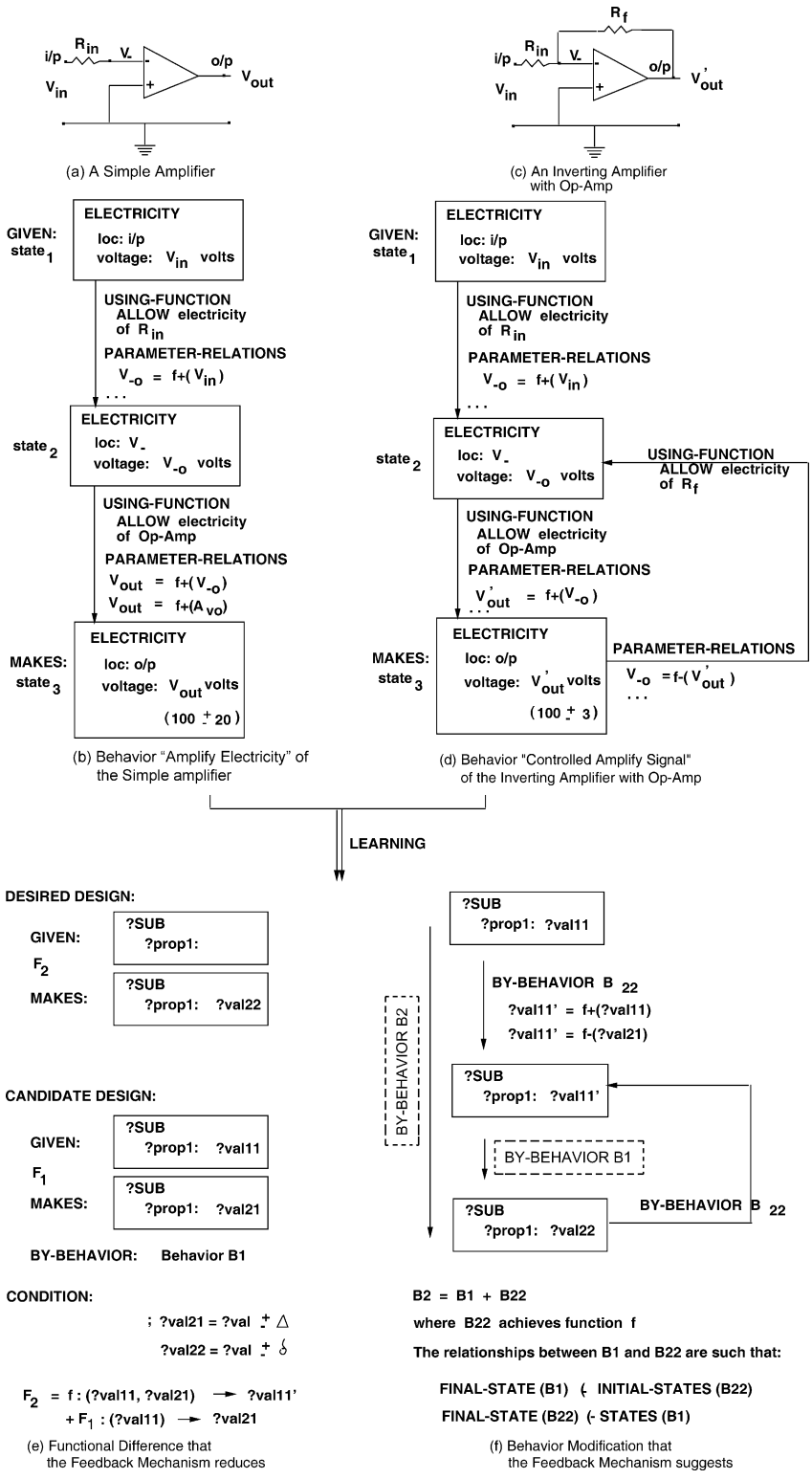

Fig. 2. Learning of feedback mechanism.

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 Fig. 2(b) shows the function 'Amplify Electricity' of the simple amplifier. Both the input state and the output state are represented as *substance schemas*. Informally, the function specifies that the amplifier takes as input electricity with a voltage of V_{in} volts (i.e. 1) at *i/p* and gives as output electricity with a voltage of V_{out} volts (i.e. 100 ± 20 where 100 is the average value and 20 is the fluctuation around the average value) at *o/p*. Note that while the representation of a specific design

may specify fluctuations in terms of quantitative tolerance limits (e.g. the voltage is 100 ± 20), the representation of a design pattern would specify fluctuations in terms of qualitative abstractions such as *small*, *medium* or *large*, independent of specific quantitative values.

The internal causal **behaviors** in the SBF model of a device explicitly specify and explain how the functions of structural elements in the device get composed into device functions. The annotations on the state transitions express the *causal*, *structural*, and *functional contexts* in which the transformation of state variables, such as substance, location, properties, and parameters, can occur. Fig. 2(b) shows the causal behavior that explains how electricity applied at the input location *i/p* of the simple amplifier is transformed at the output location *o/p*. The annotation USING-FUNCTION in the transition $state_2 \rightarrow state_3$ indicates that the transition occurs due to the primitive function 'ALLOW electricity' of operational amplifier (op-amp). The PARAMETER-RELATIONS associated with this transition $state_2 \rightarrow state_3$ indicate how the output voltage V_{out} is dependent on the voltage at location V_- and the open loop gain A_{Vo} of the op-amp. The two transitions $state_1 \rightarrow state_2$ and $state_2 \rightarrow state_3$ together explain how the input voltage V_{in} is transformed into the output voltage V_{out} of the simple amplifier and how it achieves the amplification function. The difference between the behaviors of simple amplifier and inverting amplifier is that in the latter the output voltage alters the intermediate input and thus determines the overall output voltage. In specific terms (Fig. 2(d)), the voltage at location V_- is not only dependent on V_{in} but also on the output voltage V_{out} that is fed back to the location V_-, and the voltage at V_- in turn determines the overall output voltage V'_{out} of the inverting amplifier.

In addition to representing structure, behavior, and function, an SBF model organizes this knowledge in a F→ B→ F→ B→ … → F(S) schemata. That is, the specification of a function of a device acts as an index to the internal causal behavior responsible for achieving it; some of the state transitions in the behavior may be annotated by USING-FUNCTION, which points to the specification of a function of some subdevice; the specification of this function of the subdevice acts as an index to the internal causal behavior responsible for accomplishing it, and so on. This F→ B→ F→ B→ … F decomposition may go on to as many levels as required by the design problem solver; at the lowest level of the decomposition, the function of the component is achieved by that component without any further specification of the internal causal behavior responsible for it.

2.2. Design patterns

IDeAL represents GTMs as behavior–function (BF) models using a subset of the SBF language as above. The SBF representation of a GTM encapsulates two types of knowledge: knowledge about the patterns of differences between the functions of known designs and desired designs that the GTM can help reduce; and knowledge about patterns of modifications to the internal causal behaviors of the known designs that are necessary to reduce the differences. That is, it specifies relationships between patterns of functional differences and patterns of behavioral modifications to reduce those functional differences. For example, Fig. 2(e) and (f), respectively, show these two types of knowledge for a partial model of the feedback mechanism.[3] Fig. 2(e) shows the patterns of functions F_1 and F_2, respectively, of a candidate design available and the desired design, and the conditions underwhich the mechanism is applicable. Because of the tasks for which they are used in MBA, the GTMs are indexed by the patterned functional differences such as shown in Fig. 2(e) (i.e. the fluctuations in the output substance property values in the candidate design function and the desired design function, respectively, are large and small as denoted by Δ and δ around an average value ?*val*). The model of the 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 (i.e. B_1) and the additional behavior B_{22} (that achieves the subfunction of F_2 in addition to F_1 characterized as *f* in the conditions of the mechanism). In particular, the feedback mechanism suggests the addition of a causal link (as indicated by B_{22}) from a change in the output substance state to a change in an earlier state (input state or intermediate state) in the candidate behavior so that the effective input to the device is modified. Fig. 2(f) shows the relationships in the model of the feedback that IDeAL learns from the two designs of amplifiers.

In more specific terms, the feedback mechanism as represented in Fig. 2(e) and (f) is an abstraction of similarities and differences between the SBF models of the simple amplifier (Fig. 2(b)) and the inverting amplifier (Fig. 2(d)). It suggests that one can achieve the function F_2 (?*val*11 → ?*val*22 where ?*val*22 fluctuates small) by composing the behavior B_1 of a similar function F_1 (?*val*11 → ?*val*21) and the behavior B_{22} that transforms ?*val*11 into the intermediate input *val*11′ by taking a sample of ?*val*21 (the output of F_1 which fluctuates large) so that the modified intermediate input *val*11′ in turn produces the overall desired output of ?*val*22. Note that *val*11′ is dependent both on the input *val*11 and the output *val*21 (which fluctuates large and which is fed back). The resulting, overall output of B_2, however, will be *val*22 (which fluctuates small).

3. Acquisition of GTMs

Let us consider the situation in which IDeAL's analogue memory contains the design of a simple amplifier (Fig. 2(a)

[3] Control can be open loop or closed loop. The feedback mechanism described here is one type of closed-loop control in which the output substance, feedback substance, and the input substance are all same.

and (b)). Note that the output of this device is dependent on the open loop gain (A_{Vo}, a device parameter) of the op-amp and is typically very high (ideally ∞) and unstable. Consider the scenario where IDeAL is given the problem of designing an amplifier whose function is to deliver a specific, controllable output electricity—an output which does not fluctuate much. That is, an output electricity with a voltage value, V'_{out} volts (i.e. 100 ± 3 where 100 is the average value and 3 is the fluctuation allowed around the average value) given an input electricity of V_{in} ($=1$) volts. See MAKES and GIVEN states in Fig. 2(d). IDeAL uses the specified function as a probe into its memory of analogues and retrieves the design of the simple amplifier (Fig. 2(a) and (b)) because the two functions are similar (i.e. the input states are identical and the output states differ only in a parameter value and a constraint on that value). Suppose now that IDeAL only has a simple strategy such as replacing a component in a past design to deliver new functions. Given the model of the simpler amplifier as shown in Fig. 2(b), IDeAL cannot localize the needed modification to reduce the difference between the source and the target and hence it fails. Under this failure, if an oracle presents the correct design (the diagram of the structure of the new device is shown in Fig. 2(c)) and the SBF model of the new device (shown in Fig. 2(d)), IDeAL learns a model of the feedback mechanism (shown in Fig. 2(e) and (f)) that encapsulates the abstracted functional differences and the abstracted behavioral relationships. IDeAL learns the feedback mechanism from the two designs of amplifiers (one without feedback and the other with feedback) by differential diagnosis and model-based learning: given the SBF models of the two designs of amplifiers in a problem-solving context, IDeAL focusses on the parts relevant to the context and compares them state-by-state to determine what differences in the device behaviors might be responsible for the functional differences; once it identifies them, it abstracts the behavioral and functional relationships from the device-specific models to form the generic BF model of the feedback GTM shown. Bhatta and Goel [2,3] provide a detailed account of the learning task and method.

4. Analogical transfer based on GTMs

Let us now consider a design problem in the domain of mechanical controllers presented to IDeAL. The new problem has a functional specification that given the substance angular momentum with a magnitude of L_i and clockwise direction at an input location (gyroscope), the device needs to produce the angular momentum with a magnitude L'_o proportional to the input and the same direction at a specified output location. It also specifies the constraint that the output cannot fluctuate much around an average value (i.e. $L'_o = L_{avg} \pm \delta$, where δ is small). This is the problem of designing a gyroscope follow-up [14].

Let us consider the knowledge condition in which the design of a device (Fig. 3) which transfers angular momentum from a gyroscope to an output shaft location is available in IDeAL's analogue memory (or is given explicitly as part of the *adaptive* design problem). Given an input angular momentum of magnitude L_i and clockwise direction at the input (gyroscope) location, this device produces a proportional angular momentum of magnitude L_o and of clockwise direction at the output shaft location; however, L_o fluctuates over a large range, i.e. $L_o = L_{avg} \pm \Delta$, where Δ is large. Fig. 3 shows the design of the available device: its function, structure and internal causal behavior. IDeAL retrieves (if not given explicitly) the design of

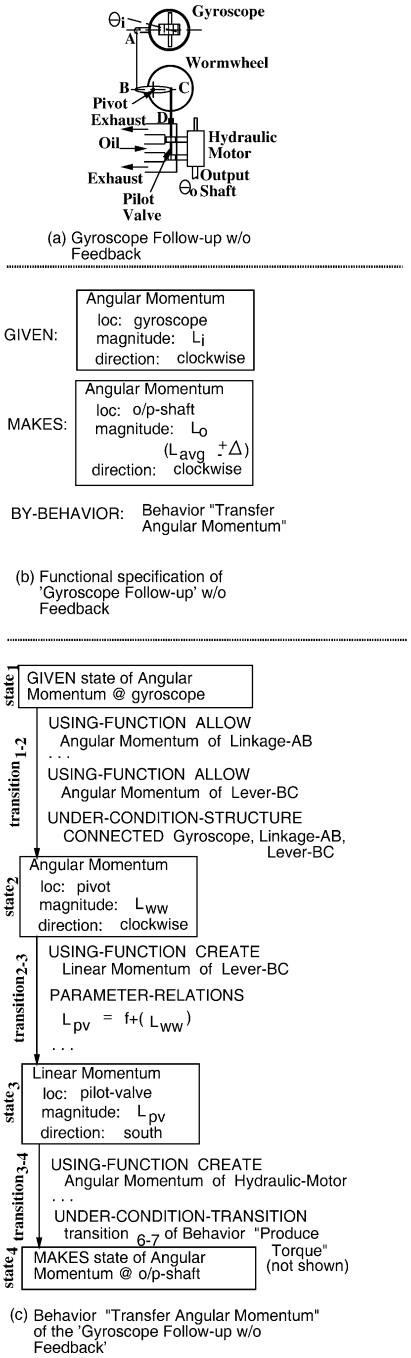

Fig. 3. Simple gyroscope follow-up (without feedback control).

gyroscope control system available in its memory because the desired function matches with the function of this design. That is, the function of this available device is similar to the function of the desired design in that the input states are identical and the output states differ in a parameter value (i.e. magnitude of angular momentum) and a constraint on that value.

Now, the task for IDeAL is to modify the available design of gyroscope control system to deliver the desired function. IDeAL attempts to use the SBF model of known design in the manner of Kritik [12], but fails because it cannot identify any component in the design that can be modified to achieve the desired function. The issue now becomes if and how IDeAL can automatically modify the device topology using the knowledge of GTMs.

IDeAL first retrieves the relevant GTM from its memory: it uses the difference in the functions of the candidate and desired designs as a probe into its memory because it indexes the mechanisms by the functional differences and the decomposability conditions on the desired functions. It retrieves the feedback mechanism because the current functional difference, namely, the fluctuation in the output property being large vs. small (i.e. Δ vs. δ), is same as the difference that the feedback mechanism reduces which is specified in a device-independent manner. Then, it tries to match the decomposability condition on the desired function in the feedback mechanism (see Fig. 2(e) for the condition $F_2 = f$: $(?val11, ?val21) \rightarrow ?val11' + F_1$: $(?val11) \rightarrow ?val21$) with the desired function in order to find the subfunctions f (or g) that need to be designed for and composed with the candidate function. By performing this match, as guided by the language of SBF models, IDeAL finds the subfunction f:$(L_i, L'_o) \rightarrow L'_{ww}$, i.e. it needs to design for a structure that takes two inputs, angular momentum with a magnitude of L_i and angular momentum with a magnitude of L'_o, and gives as output an angular momentum of L'_{ww} in the opposite direction at the location of pivot in the candidate design.

Next IDeAL instantiates the retrieved GTM in the context of the target problem. The algorithm for IDeAL's process of instantiating the GTMs is shown in Fig. 4. When the abstractions are GTMs, this process involves designing for the subfunction(s) determined by matching the applicability conditions of the mechanism (in steps 2 and 3 of the algorithm) and composing the new sub-behavior(s) with the behavior of the candidate design as per the relationships specified in the retrieved mechanism (in step 5). Let us walk through the algorithm as it applies to the current example. Step 1 is to select the behavior relevant for the function of the available design. Since the function in an SBF model of a device directly points to the behavior relevant for that function, this step is trivial.

Input: • M_1, the SBF Model of the Design Analogue, and its Function F_1.
• F_2, the desired function.
• G, a GTM (retrieved by matching $F_2 \sim F_1$).
Output: • M_2, the SBF Model of the new device that achieves F_2.
Procedure:
begin
(1) Select the behavior B_1 in M_1 relevant to F_1.
(2) Bind the initial & final states of B_1 to the appropriate GIVEN and MAKES states of the subfunctions f and g in G.
(3) **if** ∃ an unbound state variable in f or g
then backtrace B_1 to find states in B_1 that may be modified, considering the bindings from step **2**.
(3.1) **if** ∃ multiple candidate states for modification
then Select the state that is nearest to the final state in B_1.
(3.2) Compute values of unbound state variables in f and g based on the selected state, $(F_2 \sim F_1)$, and PARAMETER-RELATIONS in B_1.
(4) **if** ∃ multiple GIVEN or MAKES states in f or g
then Check if ∃ $b \in B_1$ that achieves the transformation from any of the GIVEN states to any of the MAKES states in f or g.
(4.1) **if** *yes*
then f' = rest of the transformation in f.
(i.e., < (GIVEN-states(f) - initial-state(b)),
(MAKES-states(f) - final-state(b)) >.)
g' = rest of the transformation in g.
(i.e., < (GIVEN-states(g) - initial-state(b)),
(MAKES-states(g) - final-state(b)) >.)
(5) Retrieve subdesigns for f' and g'.
(5.1) **if** ∃ no subdesigns for f' or g' **then** FAIL.
(5.2) **else**
(5.2.1) Adapt the retrieved subdesigns for f' and g' (if necessary).
(5.2.2) Compose $B_{f'}$, the behavior for f', and $B_{g'}$, the behavior for g', with B_1 as per the relationships in G.
(5.2.3) Propagate the resulting changes in state variables forward in B_1 and in the dependent behaviors in M_1.
end

Fig. 4. IDeAL's method for instantiating a GTM.

In the current example, B_1 is the behavior shown in Fig. 3(c). Step 2 is to identify bindings for variables in the retrieved GTM, in particular, in the subfunctions to be designed for. Some of the bindings for the state variables are obtained while doing the matching for the retrieval of the GTM itself. As described above, in the current example, IDeAL finds the subfunction f to be $(L_i,L'_o)\rightarrow ?val11'$ because $?val11$ is the value of the property (whose output values in the desired and retrieved functions are different) in the initial state of B_1 and $?val21$ is the value in the final state of B_1. Like in this example, even after step 2, some other variables such as $?val11'$ still need to be bound with specific values from the behavior of the available design. Step 3 is exactly for doing that: the idea is to trace the relevant behavior of the available design, B_1, backwards from the final state to the initial state, and identify the intermediate states that are possible candidates for the states of the subfunctions. In the current example, IDeAL needs to find a candidate state from B_1 that could be the output state of the subfunction f. As it traces back the behavior shown in Fig. 3, the first state to be considered is $state_3$. But since it describes a substance (linear momentum) different from what the substance (angular momentum) is from the bindings in step 2, this state cannot be a candidate. Next, it considers $state_2$ which is the only state left and which is a candidate for the output state of f. If $state_3$ were to describe angular momentum, it would also have been a candidate. In such a case, IDeAL would have chosen $state_3$, the state nearest to the final state of B_1. The rationale in this is that the modification selected should cause as minimal disturbance in the candidate behavior as possible, which means modifying the state as near to the final state as possible in order to solve the problem. Since $state_2$ is selected in the current example, we get the binding for $?val11'$ from this state, and we have all parts of the subfunction specified.

Since the subfunction has multiple states, step 4 is relevant. In the current design scenario, the subfunction IDeAL needs to design really has two parts (as it takes two inputs and produces one output): one that specifies the need for transferring angular momentum from the input location to the pivot location, and the other for transferring angular momentum from the output shaft location to the pivot location. Applying step 4, we can find that $transition_{1-2}$ really covers the transformation $?val11\rightarrow ?val11'$ and the remaining transformation (f') in f is $?val21\rightarrow ?val11'$. That is, the first part is already designed for in the candidate design as the behavior segment $state_1\rightarrow state_2$ (Fig. 3(c)) achieves it. Therefore, in successfully instantiating the mechanism in the candidate design of gyroscope follow-up, IDeAL only needs to find a behavior (and a structure) that accomplishes the second part of the subfunction (f') given the context of the first transformation. Note that while a complex device may have many subfunctions, since GTMs are high-level abstractions, a GTM is likely to have only a small number of subfunctions.

Let us consider the knowledge condition in which IDeAL has the knowledge of a component (called *worm*) whose function is to transfer an input angular momentum to an output location with the magnitude proportional to the output component and the direction dependent on the direction of threading on the worm. This component reverses the direction of the input angular momentum. In step 5, given the subfunction f', IDeAL retrieves that component because the desired part of the subfunction matches with the component's function. It substitutes the appropriate parameters in the behavior of the retrieved design (i.e. worm) to generate a behavior for the desired subfunction. Then it composes that behavior (i.e. B_{22}) with the behavior of the candidate design (i.e. B_1) as per the specification of the causal relationships in the feedback mechanism (as in Fig. 2(f)) to propose a behavior (shown in Fig. 5(c)) for achieving the desired function. Note that the resulting modification is non-local in that it modifies the device topology. It finally propagates the changes in states resulting from composing the subdesign's behavior with B_1 forward to the final state or until a state is revisited.[4]

Finally, IDeAL evaluates the candidate design by qualitative simulation of the model it generates, where the qualitative simulation is done simply by tracing the causal behaviors and propagating the effects of the initial conditions and constraints imposed by transitions.

5. Evaluation

IDeAL provides a testbed for experimenting with the MBA theory. We conducted several kinds of experiments with IDeAL that evaluate the MBA theory for its acquisition, access, and use of GTMs. Each experiment typically contained two steps. The first step involved giving IDeAL a pair of designs, one without any instance of GTM and the other with an instance of a GTM, and testing IDeAL's ability to learn a BF representation of the GTM instantiated in one of the two input designs. In the second step, IDeAL is given a design problem, from a different domain in some cases, such that it would need to access and use a previously learned GTM in order to solve the given problem. We verified if it can autonomously recognize the applicability of a GTM and successfully access and use it to solve the given problem. We conducted 12 such experiments with different combinations of design sources and target problems from four different design domains involving 28 distinct designs. The largest design contained about a dozen components.

As mentioned above, one of strengths of the SBF models is that it organizes knowledge of the structure, behavior, and

[4] In general, of course, multiple designs may be retrieved from memory for f' or g', and thus IDeAL would need to select among the retrieved designs. The current version of IDeAL assumes that only one design is retrieved for f' and one for g'.

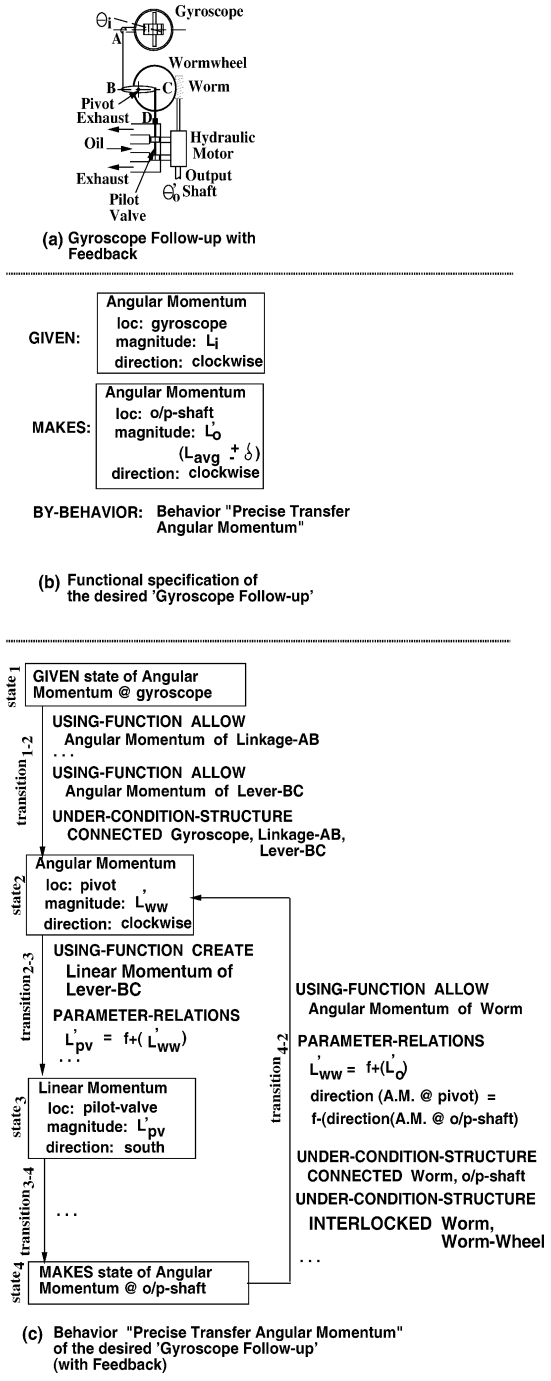

Fig. 5. Gyroscope follow-up instantiating the feedback mechanism.

function of a device in a (F→ B→ F→ B→ … F→ B→ F(S)), hierarchy, which breakdowns a potentially very large behavior into a number of smaller behaviors, which are easier to search. The largest SBF model of a device in IDeAL contained three levels of organization (F→ B→ F→ B→ F(S)), and the largest graph in a behavior contained about a half-dozen nodes. We tested IDeAL's learning of six different GTMs, four of which were related to control systems, and its use of three of them. In all these cases, IDeAL was successful in learning GTMs, and in accessing and using them in solving design problems. The behavior of IDeAL in these experiments led us to conclude the results in the following four dimensions:

(1) *Computational feasibility and efficacy*. IDeAL successfully addresses the multiple tasks in the MBA theory, for example, the tasks of learning, accessing, transferring and using GTMs. It thus demonstrates the computational feasibility and efficacy of the MBA theory.
(2) *Uniformity of representations*. The different tasks in the MBA theory impose constraints on one another. They also impose different constraints on the design knowledge representations. IDeAL uses the same SBF language for addressing the different tasks. The GTMs, for example, are represented in a BF language that is a subset of the SBF language. The design analogues too are indexed in the SBF vocabulary.
(3) *Generality of domains*. As mentioned above, IDeAL presently contains about 30 design analogues from four different device domains, namely, the domains of simple electric circuits, heat exchangers, electronic circuits, and complex mechanical devices (such as momentum controllers and velocity controllers). This includes the design problem used as an illustrative example in this paper, which was taken from a classical textbook on mechanical design [14].
(4) *Generality in terms of different GTMs*. IDeAL presently covers six different GTMs: cascading, four different types of feedback, and one type of feedforward.

6. Related research

Design patterns are often discussed in the design literature as a basic unit of design knowledge. Gamma et al. [9], for example, have advocated the construction of libraries of design patterns for supporting interactive object-oriented programming. Implicit in their work is the notion that design patterns form productive units of reusable design knowledge. Our work makes this idea explicit: design patterns are productive units of analogical transfer (and, thus, reuse) of design knowledge.

SYN [4] and DSSUA [16] are two examples of analogical design systems (in contrast to case-based design). The former is a module within the FABEL system, a CAD environment intended to assist human architects in designing spatial layouts of air-circulation systems in buildings. It represents its design cases as graph-based topological models. The models are organized in a tree of design concepts, where a design concept corresponds to a maximal common subgraph between the designs under the concept. SYN's design concepts may be viewed as topological design patterns. But it does not abstract any pattern from specific designs but rather assumes the patterns as given. Also, its use of the patterns is limited to simple within-domain analogies.

DSSUA is based on the notion of design prototypes. Like design patterns, design prototypes too specify functional relations and causal structures in a class of devices, but, unlike a design pattern, a design prototype also specifies the generic physical structure of the device class. While a design prototype is a generalization over design cases such that a case is an instance of a prototype, a design pattern is an abstraction over design prototypes such that a design prototype is a subclass of a design pattern. DSSUA uses an analogical process similar to that of the structure-mapping engine (SME) [7] to abstract causal behaviors at transfer time. More recently, Qian (2002) [17] has presented a general theory of analogy-based design based on the DSSUA system. In contrast, in MBA, design patterns are abstracted at storage time and acquired for potential reuse. The design patterns are indexed by the problem-solving goals stated in terms of functional differences between two design situations. This aspect of MBA is similar to purpose-directed analogy [15].

Faltings (2002) [8] describes the FAMING system that uses a geometric representation of designs and performs SBF analysis on them. FAMING is concerned with the creation of new designs of kinematic pairs. It begins a metric diagram of a familiar case, where a metric diagram is a line drawing consisting of polygons along a specification of constraints and variables (angles, rotations, distances). It uses the metric diagram to deduce a configuration space for the kinematic pair, a qualitative behavior (i.e. an ordered sequence of kinematic states), and a functional interpretation of the behavior. The design is adapted by a structure–behavior inversion, whereby the functional specification is matched to specific behavioral instances and qualitative places.

Case-based theories of design mostly involve direct transfer of the structure of familiar designs to new design situations. That is, the transfer is not mediated by high-level abstractions. In some case-based theories (e.g. [12]), however, high-level abstractions do enable case reminding, but still play little role in analogical transfer. Also, the high-level abstractions in these case-based theories are generalizations over features of a problem, and do not specify relations that characterize a problem and its solution. Finally, design adaptations in case-based design are in general limited to local (typically parametric) design modifications.

The origin of our structure–behavior–function (SBF) models lies in Chandrasekaran's functional representation (FR) scheme for representing the functioning of devices [6,20]. Thus, SBF share the main features of FR: (i) functions of devices are represented explicitly, (ii) functions act as indices into internal causal behaviors responsible for them, (iii) behaviors are represented as an ordered sequence of states, (iv) state transitions in a behavior are annotated by the causes for them, (v) state transitions as indices into functions at the next (lower) level of aggregation, and (vi) the states in a behavior may encapsulate other internal causal behaviors. The main differences between SBF and FR are that (a) SBF models admit specification of all output behaviors of a device, where the functions of a device are a subset of its output behaviors, (b) SBF models use a component-substance-field ontology of devices, which enables a more precise specification of states in a behavior or in a function, (c) SBF models use an ontology of primitive functions [5], which enables a more precise specification of state transitions in a behavior, (d) in SBF models, separate behaviors are constructed for substances, components and fields, which makes for a more precise specification of behaviors as a whole, (e) the internal causal behaviors in an SBF model may branch and merge, and (f) the internal causal behaviors admit inverse causality and bi-directional causality. While (a), (e), and (f) above enhance the expressive power of SBF models, (b), (c), and (d) afford more precise and accurate inferences; the enhanced expressive power expands the domain generality of SBF models and the precision of SBF models expands their task generality.

Rasmussen [18] proposed a different theory of structure–behavior–function models to support interactive operation and trouble-shooting of complex physical systems. Umeda et al. [23] describe another theory of function–behavior–structure models to support interactive design of complex physical systems. Although these various theories differ in some ways, a common theme in them is that behavior mediates between function and structure. More recently, Sasajima et al. [19] described function–behavior representation language (FBRL) and its use for explaining the functioning of devices. While a device function in our SBF models is an abstraction of a device behavior, a function in FBRL corresponds to a state that is the goal of the behavior.

The MBA theory evolves from an earlier theory of case-based design called *adaptive modeling* [10,11]. The adaptive-modeling theory described case-specific structure–behavior–function (SBF) models. It showed how case-specific SBF models enable local (i.e. parametric or componential) modifications to source designs for solving target design problems. It also showed how case-specific SBF models of new designs can be acquired by adapting the models of known designs. Stroulia and Goel [22] described how case-independent generic models enable topological modifications to source designs in the same domain as that of target problems. Bhatta and Goel [2,3] showed how generic models of teleological mechanisms (GTMs) and physical processes (GPPs) can be acquired by abstraction over case-specific SBF models. The MBA theory completes the circle by showing how generic models mediate analogical transfer of design knowledge from the source domain to a target domain (e.g. from amplifiers to gyroscopes).

7. Discussion

That analogy plays an important role in design is a standard cliche in design research. MBA provides

a normative computational theory of analogical design based on the notion of design patterns. In the domain of physical devices, it shows that functional and causal design patterns, called generic teleological mechanisms (GTMs), constitute productive units of analogical transfer of design knowledge. It also shows that structure–behavior–function models of specific devices enable the acquisition of GTMs, that problem-solving goals pertaining to the adaptation of a familiar design to meet new functional requirements access relevant GTMs, and that the instantiation of the GTMs in the context of a partial design enables its completion.

Of course the IDeAL system is only a 'proof-of-concept' demonstration of the MBA theory. Conversion of IDeAL into a practical system for engineering design would require addressing two major issues: generality and scalability. Let us examine the issue of generality first. On one hand, we have by now used the SBF methodology and language for modeling many (a few dozen) physical systems, devices and mechanisms from several (half-dozen) domains. But, on the other, IDeAL deals with only half-dozen GTMs, four of which pertain to control systems. Further, on one hand, IDeAL has shown that SBF models enable the abstraction of generic models of teleological mechanisms and physical processes (GTMs and GPPs). But, on the other, SBF models enable the abstraction of only those abstractions that can be represented as BF models. Thus, there is a need to (i) verify that IDeAL's method of abstraction covers a large class of GTMs and GPPs, and (ii) develop a theory that covers a much larger class of abstractions.

The issue of scalability raises similar questions. On one hand, since an SBF model organizes knowledge of the structure, behavior, and function of a device in a ($F \rightarrow B \rightarrow F \rightarrow B \rightarrow \ldots F \rightarrow B \rightarrow F(S)$), hierarchy, it breakdowns potentially large behaviors into a number of smaller behaviors, and thus partitions large problem spaces into a number of smaller spaces which are easier and faster to search. On the other, the larger device we have modeled so far contained only about a dozen components. Thus, there is a need to (a) empirically verify IDeAL can handle much larger devices, and (b) formally analyze the computational characteristics of MBA. Therefore, while MBA provides an empirical and normative theory of analogy-based design, IDeAL clearly would require much more work before it can be used for supporting practical design.

References

[1] Alexander C. Notes on the synthesis of form. Cambridge, MA: Harvard University Press; 1964.

[2] Bhatta S, Goel A. From design experiences to generic mechanisms: model-based learning in analogical design. Artif Intell Eng Des Anal Manufact 1996;10:131–6.

[3] Bhatta S, Goel A. Learning generic mechanisms for innovative strategies in adaptive design. J Learn Sci 1997;6:4.

[4] Borner K, Pippig E, Tammer E, Coulon C. Structural similarity and adaptation. In: Proceedings of European workshop on case-based reasoning, Lausanne, Switzerland; 1996. p. 58–75.

[5] Bylander T. A theory of consolidation for reasoning about devices. Int J Man–Machine Studies 1991;35(4):467–89.

[6] Chandrasekaran B, Goel A, Iwasaki Y. Functional representation as design rationale. IEEE Comput 1993;48–56.

[7] Falkenhainer B, Forbus K, Gentner D. The structure-mapping engine: algorithm and examples. Artif Intell 1989;41:1–63.

[8] Faltings B. FAMING: supporting innovative design using adaptation-a description of the approach, implementation, illustrative example and evaluation. In: Chakrabarti A, editor. Engineering design synthesis. London: Springer-Verlag; 2002. p. 285–302.

[9] Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Reading, MA: Addison-Wesley; 1995.

[10] Goel A. A model-based approach to case adaptation. In: Proceedings of the 13th annual conference of the cognitive science society, Chicago; 1991. p. 143–8.

[11] Goel A. Model revision: a theory of incremental model learning. In: Proceedings of the eighth international conference on machine learning, Chicago; 1991. p. 605–9.

[12] Goel A, Bhatta S, Stroulia E. Kritik: an early case-based design system. In: Maher ML, Pu P, editors. Issues in case-based design. Hillsdale, NJ: Erlbaum; 1997.

[13] Griffith T. A generative theory of scientific modeling. PhD thesis. Atlanta, GA, USA: College of Computing, Georgia Institute of Technology; 1998.

[14] Hammond PH. Feedback theory and its applications. London: The English Universities Press Ltd; 1958.

[15] Kedar-Cabelli S. Toward a computational model of purpose-directed analogy. In: Michalski R, Carbonell J, Mitchell T, editors. Machine learning II: an artificial intelligence approach. Los Altos, CA: Morgan Kaufmann; 1988. p. 284–90.

[16] Qian L, Gero J. A design support system using analogy. In: Gero J, editor. Proceedings of the second international conference on AI in design. Dordrecht: Kluwer Academic Press; 1992. p. 795–813.

[17] Qian L. Creative design by analogy. In: Chakrabarti A, editor. Engineering Design Synthesis. London: Springer-Verlag;, 2002. p. 245–69.

[18] Rasmussen J. The role of hierarchical knowledge representation in decisionmaking and system management. IEEE Trans Syst Man Cybernet 1985;15:234–43.

[19] Sasajima M, Kitamura Y, Ikeda M, Mizoguchi R. Fbrl: a function and behavior representation language. In: Proceedings of IJCAI-95; 1995. p. 1830–6.

[20] Sembugamoorthy V, Chandrasekaran B. Functional representation of devices and compilation of diagnostic problem-solving systems. In: Kolodner J, Riesbeck C, editors. Experience, memory and reasoning. Hillsdale, NJ: Lawrence Erlbaum; 1986. p. 47–73.

[21] Shinn HS. Abstractional analogy: a model of analogical reasoning. In: Kolodner J, editor. Proceedings of the DARPA workshop on case-based reasoning, Clearwater Beach, FL, 1988. p. 370–87.

[22] Stroulia E, Goel A. Generic teleological mechanisms and their use in case adaptation. In: Proceedings of the 14th annual conference of the cognitive science society, Bloomington, IN; 1992. p. 319–24.

[23] Umeda Y, Takeda H, Tomiyama T, Yoshikawa H. Function, behavior and structure. In: Proceedings of fifth international conference on applications of AI in engineering, vol. 1; 1990. p. 177–93.