**A PRELIMINARY INVESTIGATION OF
DESIGN SIMPLIFICATION BY ANALOGY**

M.E. BALAZS, D.C. BROWN
AI in Design Group
Computer Science Department
Worcester Polytechnic Institute
Worcester, MA 01609, USA.

E-mail: mebalazs@cs.wpi.edu, dcb@cs.wpi.edu
URL: http://www.cs.wpi.edu/~mebalazs, http://www.wpi.edu/~dcb

Phone: (508) 831-5618
FAX: (508) 831-5776

A PRELIMINARY INVESTIGATION OF DESIGN SIMPLIFICATION BY ANALOGY

Abstract. This paper describes research on the simplification of designs using analogical reasoning. We present a definition of simplification and some examples, and then describe a prototype system that has been built to support our preliminary investigations of this problem.

1.0 Introduction

This research is concerned with the representations and reasoning required for Design Simplification. A design can be considered as simpler than another for a variety of reasons, such as shape, use, or ease of assembly, for example. Simplification, as a consequence, can be done with a variety of goals. In addition, simplification can be done in a variety of ways. For example, simplifications might be searched for (i.e., reasoned out) or retrieved (e.g., using CBR).

Our preliminary investigations are concerned with doing simplification by Analogy with stored simplifications.

The designs are represented as Function, plus Behavior, plus Structure. Structure is represented in terms of attributes, components and relations between components. Behavior is represented as state transition graphs, and causal chains. Function is represented as sets of inputs and outputs, with possible additional annotations about class of use, physical principles [Bhatta et al. 1994], or functional primitives [Keuneke 1991]. All simplifications are supposed to preserve the artifact's intended function.

Simplifications may be made at each of these levels, with possible consequences for the other levels of the representation. For example, simplifying the behavior may make a component redundant, leading to a structural change.

The propagation of changes across levels of representation (e.g., from behavior to structure) is, as far as we know, a new research area. The design simplification problem area in general raises many interesting research issues concerning functional reasoning and representations, and their interaction with analogical reasoning. In addition to the obvious benefits of any new research results, better understanding of the simplification of designs might eventually lead to the important consequence of cheaper, better designed products.

We do not know of any ongoing research in the area of "goal-based analogical reasoning for design simplification". However, there is certainly a rich body of research results in design simplification, functional reasoning, analogical reasoning in general, and purpose-directed analogical reasoning in particular.

A general approach to design simplification is Suh's [1990] "Reduction of the Information Content of a Product". The Design For X (where X = Manufacturing, Assembly, Recycling, etc.) research community define clear principles and methods for reducing the life-cycle costs of a product [Boothroyd & Dewhurst 1991]. Reasoning about designs from different points of view (view-point-based abstraction) is discussed by Manfaat [1997] [Manfaat et al 1996].

The main results in the domain of functional reasoning refer to defining functional representation schemes [Sembugamoorthy & Chandrasekaran 1986] [Chandrasekaran 1994] [Goel 1992] [Umeda & Tomiyama 1994] [Chittaro et al. 1993], reasoning about function [Thadami & Chandrasekaran 1993] and applications of functional reasoning, such as conceptual design [Umeda & Tomiyama 1994], diagnosis [Chittaro 1995], and blame assignment [Stroulia & Goel 1995].

The main directions of research in analogical reasoning are the study of analogical reasoning as a cognitive process and as one of the main ingredients of creativity [Boden 1994] [Holyoak and Thagard 1995] [Finke et al. 1992], types of analogical reasoning (e.g., transformational versus derivational) [Gentner 1993] [Falkenheimer et al. 1993] [Carbonell 1993] and applications of analogical reasoning to learning [Carbonell 1993], planning [Veloso 1994], design [Qian & Gero 1992] [Bhatta & Goel 1994] [Bhatta et al.] [Goel 1997]. Of special interest for our work is the research in purpose-directed analogy [Kedar-Cabelli 1988].

1.1 Example Domain

To illustrate the concepts we are going to use, and to demonstrate our preliminary experimental results, we shall use examples from the domain of door lock designs [Chakrabarti & Tang 1996].

We consider a door lock to be a device that allows and prevents the opening of a closed door (or gate or window). It is composed of a *box* and a *bolt* which can be fully retracted into the box as a consequence of some input applied (Figure 1a). When the bolt is completely retracted it allows the door (together with the whole lock) to move into the open position. Note that in order to simplify things we are only concerned here with the unlocking behavior of the lock.

For our purposes, the behavior of the door lock is a three state process (Figure 1b). The initial state corresponds to the closed door and is characterized by an unretracted bolt and a shut door. The second (intermediate) state is characterized by a retracted bolt and a shut door. The door lock may get into this state from the initial state as a consequence of applying some input. The final state corresponds to the open door and is characterized by a retracted bolt and an open door. This state can be reached from the intermediate state by applying a second input to the door lock.

In our examples we shall limit ourselves to door locks implemented using simple components with distinct functional roles (such as levers, cams, shafts, etc.). We shall also assume that the inputs to a door lock will be forces characterized by their directions. Figures 2 through 4 present schematic examples of door locks.

2.0 Design Simplification

2.1 WHAT IS SIMPLIFICATION

By a “design” we mean a *description* that is the result of some design activity, where that description is realizable as an engineered artifact (i.e., it can be made and tested). A description is factored into *levels* that describe the designer-intended structure, behavior and function of the artifact. A design simplification can be made by changing one or more of these levels. A simplification results in a different design.

Whether such a change is actually a “simplification” or not can be evaluated by computing (or physically testing) its consequences in some context, according to some measure. Contexts include assembly, manufacturing, use, aesthetics, etc. Measures involve counting, or some measure of complexity or information content. Lower counts, complexity or information content imply ‘simpler’. What is counted depends on the context: for example, it may be assembly operations, components, surfaces, the potential for manufacturing mistakes, internal states, inputs, or the number of terms in an equation that describes a surface.

2.2 TYPES OF SIMPLIFICATION

The basic views of simplification are at each of the three levels of description. For example, a change made to the structural level that results in a simplification is referred to as a ‘structural simplification’. However, this is really just a ‘convention’, as simplifications, while made on descriptions, actually take effect in a particular context/measure setting.

A *structural simplification* refers to either physical attributes of the artifact or to its structural composition. For instance, an object with the shape attribute “sphere” may be considered simpler than an object with the shape attribute “oval”. With respect to structural composition, a design can be simpler than another design if it has fewer components, fewer relations between components, or simpler relations between components. Here “simpler relations” itself requires a measure (e.g., fewer constraint on those relations).

A *behavioral simplification* occurs if the new design has either fewer states, or fewer transitions. Here, “fewer” might mean “subset of”, as might occur if some states were redundant and could be removed. It might also mean “smaller number”, as might occur if a behavior with different, and not so many, states were adopted. We suspect that both may produce interesting simplifications.

A *functional simplification* may refer to either the “complexity” of using a designed object, or to its possible multiple functions. An object may be “functionally simpler” than another object if it is easier to use. For example, if it requires fewer inputs or simpler inputs (e.g., force applied in a linear rather than curved motion). On the other hand, an object can also be “functionally simpler” than another object if it has fewer functions (i.e., it can be used for fewer purposes).

Other possible views of simplification are named for the *context* in which the *measure* is applied. For example, an “assembly simplification” is a simplification from the point of view of Assembly. It might, for example, reduce the number of handling operations, perhaps by adding more part symmetry.

Just to confuse things, one possible context might be the Design (i.e., the description itself) with the measure being Description Length. An interesting context is Aesthetics, where we measure the degree of our pleasurable response to the artifact’s beauty. Another interesting context is Understanding, where we might measure the difficulty in working out exactly what a design (or the corresponding artifact) is for, or how it works.

There is a tendency for people to *assume* that certain types of changes are simplifications. An example of this is the belief that ‘simpler = fewer components’. Such examples have been compiled into peoples’ knowledge by repeated use, and appear to be context-free to those individuals. However, we feel that they can be traced back to contexts such as manufacturing or use. We suspect that the common ‘assumed’ simplifications are those that lead to being evaluated as simpler in common contexts and measures, or that lead to being evaluated as simpler in the important majority of contexts and measures.

We shall illustrate the basic views of simplification using the door locks presented in Figures 2 through 4.

Note that the door locks in Figures 2a and 2b are essentially identical except for the implementation of the cam mechanism that transforms rotation into translation. In Figure 2a this mechanism is implemented using a roller, while in Figure 2b it is implemented using a cam.

For the roller implementation, the shape of both the bolt and the roller is simpler than the corresponding components in the cam implementation. Thus we can say that the roller implementation is structurally simpler than the cam one. Note however that if the cam in 2b was somehow simpler than the roller we could not rank the two designs by a simpler relation.

If we compare the number of components with functional roles used to build either of the designs in Figure 2 (lever, cam and bolt) to the number used for the design in Figure 3 (two levers, wedge and bolt) we can consider the former to be structurally simpler than the latter.

None of the door locks considered have redundant states that could be removed to produce a behavioral simplification. It is however possible to compare them from the point of view of their behavioral complexity, based on the complexity of the causal graph (e.g., the length of the causal chain) which connects the input to the output. We can say that the door lock in Figure 4 is behaviorally simpler than the one in Figure 3 because the causal chain is shorter.

The door lock in Figure 4 is simpler in *use* than all of the other door locks considered because it only requires the application of a single input which will both retract the bolt and move the door into the open position. The other door locks all require two separate, sequential inputs from different directions. We can say that the door lock in Figure 4 is functionally simpler than any of the other examples.

2.3 PROPAGATION OF SIMPLIFICATION

Simplification at one level will need to propagate to the other levels, as function depends on behavior which depends on structure. For example, removing redundant links in a causal chain of motion or force flow (i.e., simplifying behavior) might cause two gears in a three gear train to be removed.

Even though our prototype system isn't currently able to do this, it will eventually need to perform this propagation and analyze its consequences using causal reasoning. This is needed for several reasons. First, almost all simplifications of a design eventually have to be made at the structural level. Thus, even if a functional or behavioral simplification is attempted, the system will need to derive the structural changes required. Second, a simplification at one level may propagate to another level and result in a violation of some design requirements and/or constraints. In such a situation the proposed simplification must be discarded, or, possibly, be modified.

Assume that we want to perform a functional simplification on the design in Figure 2a similar to the one exhibited by the design in Figure 4, that is, to modify the design such that only a single input with the same direction as I2 is needed. To allow this we would need to replace the lever (corresponding to the door handle) by a mechanism, or combination of mechanisms, which would transform the linear movement into a rotation in a plane perpendicular to the direction of the input.

A possible implementation of this is illustrated in Figure 5. Note that while the new design is functionally simpler than the original one, structurally (as well as behaviorally) it is more complicated. Note that this is a good example of the possible need for multiple simplifications as a way if 'tidying up' a design after an initial simplification. It is also an indication of the need to make trade-offs with respect to simplicity. Something that is simpler in one context, may be less simple in other contexts. The agent controlling the whole simplification process needs to decide which context or contexts are dominant, and where trade-offs can be made.

3.0 Why Simplification by Analogy?

While some simplification rules exist for certain domains [Boothroyd & Dewhurst 1991], there is no universal classification, or set of generalizations that would lead to simplification principles.

Reusing known simplifications by adapting them to new problems allows the solution of new simplification problems. If the reuse is based on generalization and mapping to a different domain, new and previously unsolved simplification problems could be solved. More than that, the generalization would support the search for general simplification principles. These observations are the basis of our choice to use analogical reasoning to do simplification.

As Kedar-Cabelli [1988] states, analogies based only on "structure mapping" are limited in their generation and use of analogies because the *purpose* is not supplied in their input. But simplification is usually carried out to achieve some desired effect, such as lower cost or reduced time, i.e., simplification is a goal-directed activity.

In our work, the context/measure is made to serve as a "goal concept" to guide the analogical reasoning. Future work on this project will require a full study of purpose-directed analogical reasoning as it applies to the simplification problem.

4.0 A Prototype System

4.1 Representation of Designs and Simplifications

We view designs as having three important aspects: structure, behavior and function. Accordingly we represent designs at these levels.

From a *structural* point of view a design has attributes, components and relations between components. Components of a design are themselves designs and are represented as such. Designs with no components are called elementary designs. Relations between components may also have attributes.

At any moment during its use a design can be characterized by a state. A state is described by the values of the design's attribute, the states of its components and the attributes of its (internal) relations.

By a *behavior* of a design we mean the sequence of state changes through which the design goes as consequence of its set of inputs. The behavior of a design is the collection of its individual behaviors. For our purposes we only represent those behaviors of a design which are relevant to its function, to the design requirements or to design constraints.

A state transition in a behavior can be linked to a causal graph (or chain) connecting (behaviors of) its components. This results in a hierarchical description of the behavior of a design.

A *function* of a design is characterized by an intended output (or outputs) that it can produce. This intended output refers to the fact that every design is generated with an intended use. Using a design means to (purposefully) apply some inputs to it such that it produces the intended output(s). An output is a behavior (or part of a behavior) directly observable at the level of the design (rather than only at the level of its components). Based on the above, a function of a design can be described in a manner similar to behaviors, that is, as a sequence of state transitions.

The *use* of a design means the application of a set of inputs which have as consequence the realization of the function of the design.

A set of inputs to the design can be applied sequentially, simultaneously or in a combined manner. They can be represented as a precedence graph with the nodes representing the inputs and edges representing precedence relations between the application of the member inputs.

We represent the *use* of a design as a pair made up of a precedence graph representing the corresponding set of inputs, and behavior graph representing the resulting function. The nodes representing the inputs are augmented with preconditions and post-conditions for the application of that input. We illustrate our representation of designs in Figure 6. The representation given there corresponds to the door lock example in Figure 2b.

Each simplification relation (*s-relation*) contains a description of what the simplification consists of. For instance, such a description may say that the retrieved design is simpler than the one it is related to because it has fewer components. A slightly more complicated description could specify that a set of components and/or relations in the more complex design is replaced by a different but smaller set of components and/or relations.

4.2 Organizing Previous Designs and Simplifications

Our approach to design simplification is to reuse old simplifications by finding similarities to the current simplification situation and transferring these old simplification “ideas” by analogy. Clearly, our system must, in some way, represent the designs involved in the old simplifications, as well as the “simpler” relation(s) between them.

Since our purpose is to be able to perform simplification at structural, behavioral and functional levels, the s-relations must be organized into these categories.

In addition, for each level, both designs and s-relations must be organized into general-specific hierarchies. Designs are organized into hierarchies in order to allow the fast retrieval of more general, matching designs that are usable in the analogical transfer. The s-relations must be organized into hierarchies to allow fast retrieval of related simplifications, and also to allow the representation of general simplification rules.

Note that the purpose of our system does not include building design generalizations. As a consequence, the design hierarchies are considered to be given. They may be produced either by another program or by human designers.

On the other hand, s-relation hierarchies should be built, and perhaps added to by the system, with the purpose of discovering general simplifications (simplification principles). This is one of the possible contributions of our research.

4.3 Analogical Reasoning Mechanism

Matching is the mechanism by which designs similar to the design to be simplified (and their associated s-relations) are retrieved.

In what follows, by "target" design we mean the design that has been given for simplification. The existing (old) designs with which the matching is done will be called the "source" designs. We shall use these terms to indicate the roles those designs play in the analogical reasoning process.

Searching for similarities can be done at different abstraction levels, ranging from attributes and attribute values (mere appearance similarity) to relations (relational similarity).

Because we are interested in simplifying designs from different points of view we propose that matching can be done at the structural, or behavioral, or functional levels of designs.

Our approach is to search for relational similarities because they are more useful in identifying similar designs [Gentner 1988] than simple attribute or attribute value similarities. Since behavior and function are represented in the form of graphs (state transition and input precedence graphs respectively) searching for relational similarities can be done from behavioral and functional points of view as well as structural.

Note that in the literature what we call "relational similarity" is usually referred to as "structural similarity". We have avoided this terminology to prevent confusion with the structural level of description of a design.

Our implementation of matching is based on the idea of the Structure-Mapping-Engine [Falkenheimer et al. 1993]. As such it tries to find the highest level relational similarity between the target design to be simplified and the source designs considered.

Simple relational matching with respect the corresponding level of designs is inefficient because it would consider *all* the designs in the design data base. Since our goal is simplification, the matching should only be done for those designs which are involved in some simplification relation at that level. This can be further refined if the simplification relation refers to a part of design which is (in some way) similar to a part of the design to be simplified.

The matching performed by our prototype system is limited to search for s-relations of a given type, with the search criteria being the degree of relational similarity. This is the simplest possible context/measure goal concept. Our purpose is to identify and properly define context/measure goal concepts (like the ones referred to in subsection 2.1.) which would correspond to notions of simplicity as referred to in practice. In the final implementation of our simplification system such goal concepts will be used to guide the matching phase of the analogical reasoning process.

The best match found generates a mapping between some part(s) of the retrieved design and some part(s) of the target design. Based on this matching the simplification relation between the source design and its corresponding simplification must be transferred to the target design to produce a simplification of it. The description stored in the simplification relation is used to transfer the simplification from the source analog to the target.

The simplest way the transfer can be performed is to substitute the components and relations in the simplification description by the components and relations in the target design to which they were mapped to by the matching process. However, in general this transfer can be very complicated.

4.4 System Architecture

In the previous sections we presented a definition of simplification, some examples, and some requirements that a general simplification system should have. Next we describe a simple prototype system that has been built to support our preliminary investigations of this problem.

The prototype system is able to perform basic structural, behavioral and functional simplifications. We must note however that since the propagation of simplifications between levels of a design is not implemented, in the case of behavioral and functional simplifications no new design is produced, but rather a new behavioral or functional description is generated. For instance for the functional simplification presented in Subsection 2.3, the simplified function consisting of a single input would be produced. Our prototype system is not currently able propagate this to produce an actual behavioral and structural implementation for it.

The prototype system is composed of the following main parts: a data base of old designs and old s-relations, a matching mechanism for retrieving similar designs, and a mechanism for transferring simplification from an old s-relation to the target design. Currently no s-relation generalization component is implemented. However, it will be implemented in a future simplification system.

For our implementation we used the CLIPS language [CLIPS 1993]. This choice was made based on its powerful matching mechanism and its object-oriented features, which, in our view, are well suited for implementing knowledge-based systems.

The data base contains a collection of designs and s-relations implemented as objects of classes. An example of design representation is given in Figure 6. The prototype system currently uses a data base that consists of a small number of door lock designs and s-relations.

The s-relations are represented as objects of a class “relation”. They are classified into three categories, corresponding to structural, behavioral and functional simplifications. Currently this classification is represented by the value of a “type-of-simplification” attribute and the language’s matching mechanism is used to restrict matching to a given type of simplification.

The matching mechanism is implemented using the pattern matching feature of CLIPS. It has as input the target design to be simplified and the point of view from which the simplification should be done. The latter limits the set of s-relations considered to those belonging to the corresponding category.

The designs on the “less simple” side of the s-relations considered are compared to the target design. The comparison looks for a common pattern (in the topological sense) of relations in the two designs. A match gives rise to a mapping between the corresponding components. The matching mechanism considers the highest level relation pattern as describing the similarity between the two designs. For this it computes a measure of similarity which is currently just the number of components involved. A better measure must be defined in the future that collects evidence of similarity using the individual components and relations involved. The match with the highest score from all the comparisons is considered to be the best match. The s-relation corresponding to the best match is used for simplification transfer.

The transfer is implemented by applying the s-relation description to the new design. Currently, this is implemented as a simple substitution in the simplification description of the mapped components. Using this, a new design representation can be generated at the level considered. Note, however, that since no propagation has yet been implemented our prototype system does not generate a complete design representation.

4.5 An Example

To demonstrate the way our prototype system works we ran it on a set of simple examples.

The example we are going to present refers to the simplification of the design in Figure 7. Note that this design is an artificially created more complex version of the design in Figure 3.

With respect to this design we presented the system with two simplification problems, one from a structural point of view and another from a behavioral point of view. In both cases the prototype system proposed as simpler the design represented in Figure 2b. This is due to the low number of designs and s-relations represented in the data base.

The prototype system finds a match for the target design with Figure 3 because it is on the “complex” side of an s-relation of type ‘structure’. The matching mechanism finds that they have in common the “sequence” of components of types {lever, wedge}. This is the highest level of similarity found between the two designs.

The s-relation found, which connects the designs in Figure 3 and Figure 2b, contains the simplification description *(replace (sequence lever wedge l-shaped-bolt) (sequence cam notched-bolt))*. Based on this, the prototype system suggests that the {lever, wedge, l-shaped-bolt} combination in the design in Figure 7 be replaced by a {cam, notched-bolt} in order to obtain a design that is structurally simpler in a similar way that the design in Figure 3 is structurally similar than the one in Figure 2b.

For the behavioral simplification problem, the system produces the same simpler design because it is also on the “simpler” side of an s-relation of type ‘behavior’. The s-relation retrieved contains a description relating the causal chains in the two designs of the form *(replace (causal lever wedge bolt) (causal cam bolt))*. The suggested simplification can be transferred in a similar manner to the structural case.

Note that the behavioral simplification does not seem to be different from the structural one. This is because in our current representation the designs are described in terms of functional components and the causal links are placed along the structural connections between those components. There is, however, a separated description for both types of s-relations. It is quite likely that such strong couplings of structural and behavioral simplifications make it difficult, even in simplifications done by humans, to decide whether they are structural or behavioral.

5.0 Conclusions and Future Work

This paper describes research on the simplification of designs using goal-based analogical reasoning. We present a definition of simplification and some examples, and then describe requirements for a future system that incorporates our theories. A prototype system, built to support our preliminary investigations of this problem, and some results, are also presented.

Future work includes refinement of the definitions, development of a full-blown goal-based analogical reasoning mechanism for design simplification, an investigation of the use of hierarchies for intertwined designs and s-relations, and implementation of a full working system that incorporates these ideas.

This work is significant because it has the promise of developing a new type of analogical reasoning that is strongly driven by goals, demonstrating an important new application for that reasoning mechanism, providing a comprehensive definition of “simplification”, and opening the way for knowledge-based design improvement systems.

References

S.R. Bhatta, A.K. Goel & S. Prabhakar. “Innovation in analogical design: A model-based approach” In: *Artificial Intelligence in Design’94*, (Eds.) Gero & Sudweeks, Kluwer Academic Publishers, 1994, pp.57-74.

S.R. Bhatta, A.K. Goel. “Discovery of Physical Principles from design Experience”, *AI EDAM*, 2, May, 1994.

M. Boden. “What is Creativity”, In: *Dimensions of Creativity*, (Ed.) Boden, MIT Press, Cambridge, MA, London, England, 1994, pp.75-118.

G. Boothroyd & P. Dewhurst, “Product Design for Manufacture and Assembly”, In: *Design For Manufacture*, (Eds.) Corbett, Dooner, Meleka & Pym, Addison-Wesley, 1991, pp. 165-173.

J.G. Carbonell. “Derivational analogy: A theory of reconstructive problems”, In: *Readings in Knowledge Acquisition and Learning*, (Eds.) Buchanan & Wilkins, Morgan Kaufmann Publishers, San Mateo, CA, 1993, pp. 727-738.

A. Chakrabarti & M.X. Tang. “Generating Conceptual Solutions on FUNCSION: Evolution of a Functional Synthesizer“, In: *Artificial Intelligence in Design ‘96*, (Eds.) Gero & Sudweeks, Kluwer Academic Publishers, 1996, pp. 603-622.

B. Chandrasekaran. “Functional Representation: A Brief Historical Perspective”, *Applied Artificial Intelligence*, Special Issue on Functional Reasoning, 1994.

L. Chittaro “Functional Diagnosis and Prescription of Measurements using Effort and Flow Variables”, IEEE Control Theory and Applications, Vol. 142, No. 5, 1995.

L. Chittaro, G. Guida, C. Tasso, & E. Toppano “Functional and Teleological Knowledge in Multimodelling Approach for Reasoning about Physical Systems: A case Study in Diagnosis”, *IEEE Transactions on Systems, Man and Cybernetics,* Vol, 23., No. 6, 1993.

CLIPS, *C Language Integrated Production System,* Version 6.0, Lyndon B. Johnson Space Center, Software Technology Branch, 1993.

B. Falkenheimer, K. Forbus & D. Gentner. “The structure-mapping engine: Algorithm and examples”, In: *Readings in Knowledge Acquisition and Learning*, (Eds.) Buchanan & Wilkins, Morgan Kaufmann Publishers, San Mateo, CA, 1993, pp.695-726.

R. Finke, T.B. Ward & S.M. Smith. *Creative Cognition*, MIT Press, Cambridge, MA, London, England, 1992.

D. Gentner. “Analogical Inference and Access”, In: *Analogica*, (Ed.) Prieditis, Lecture Notes in Artificial Intelligence, Morgan Kaufmann Publishers, Los Altos, CA, 1988, pp.63-88.

D. Gentner. “The mechanism of analogical learning”, In: *Readings in Knowledge Acquisition and Learning*, (Eds.) Buchanan & Wilkins, Morgan Kaufmann Publishers, San Mateo, CA, 1993, pp.673-694.

A. Goel. “Representation of Design Functions in Experience-Based Design”, In: *Intelligent Computer Aided Design,* (Eds.) Brown, Waldron & Yoshikawa, North-Holland, Amsterdam, Netherlands, 1992, pp.283-308

A. Goel. “Design, Analogy and Creativity”, *IEEE Expert,* vol. 12, no. 3, May/June, 1997, pp.62-70.

K.J. Holyoak & P. Thagard. *Mental Leaps: Analogy in Creative Thought,* MIT Press, Cambridge, MA, London, England, 1995.

S. Kedar-Cabelli, “Toward a Computational Model of Purpose-Directed Analogy”, In: *Analogica*, Research Notes in Artificial Intelligence, (Ed.) Prieditis, 1988, pp.89-108.

A. Keuneke, “Device Representation: The Significance of Functional Knowledge”, *IEEE Expert,* vol. 6, no. 2, April, 1991, pp. 22-25.

D. Manfaat, The SPIDA system, personal communication, 1997.

D. Manfaat, A.H.B. Duffy and B.S. Lee, “Generalization of Spatial Layouts”, Workshop on Machine Learning in Design, 23-24 June 1996, *Fourth International Conference on Artificial Intelligence in Design’96,* Stanford University, USA, 1996.

L. Qian & J.S. Gero, “A design support systems using analogy”, In: *Artificial Intelligence in Design'92*, (Ed.) J.S. Gero, Kluwer Academic Publishers, 1992, pp.795-813.

V. Sembugamoorthy & B. Chandrasekaran. “Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Knowledge”, In: *Experience, Memory and Learning*, (Eds.) Kolodner & Reisbeck, Lawrence Erlbaum Associates, 1986.

E. Stroulia., A.K. Goel. “Representation of Design Structure, Behavior and Function for Blame Assignment”, *AI EDAM,* special issue on Functional Reasoning, 1995.

N. P. Suh, *The Principles of Design,* Oxford University Press, 1990.

S. Thadami & B. Chandrasekaran. “Structure to function reasoning”, *AAAI’93, Workshop on Representing and Reasoning With Device Function,* 1993.

Y. Umeda & T. Tomiyama. “Experimental Use and Extension of the FBS Modeler”, *AAAI’94 Workshop on Representing and Reasoning with device Function,* 1994.

M.M. Veloso. *Planning and Learning by Analogical Reasoning,* Lecture Notes in Artificial Intelligence (886), Springer Verlag, 1994.

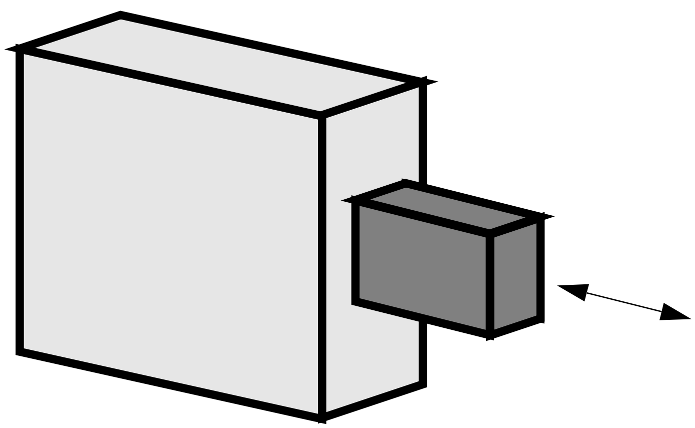

FIGURE 1a: The schematic image of a door lock.

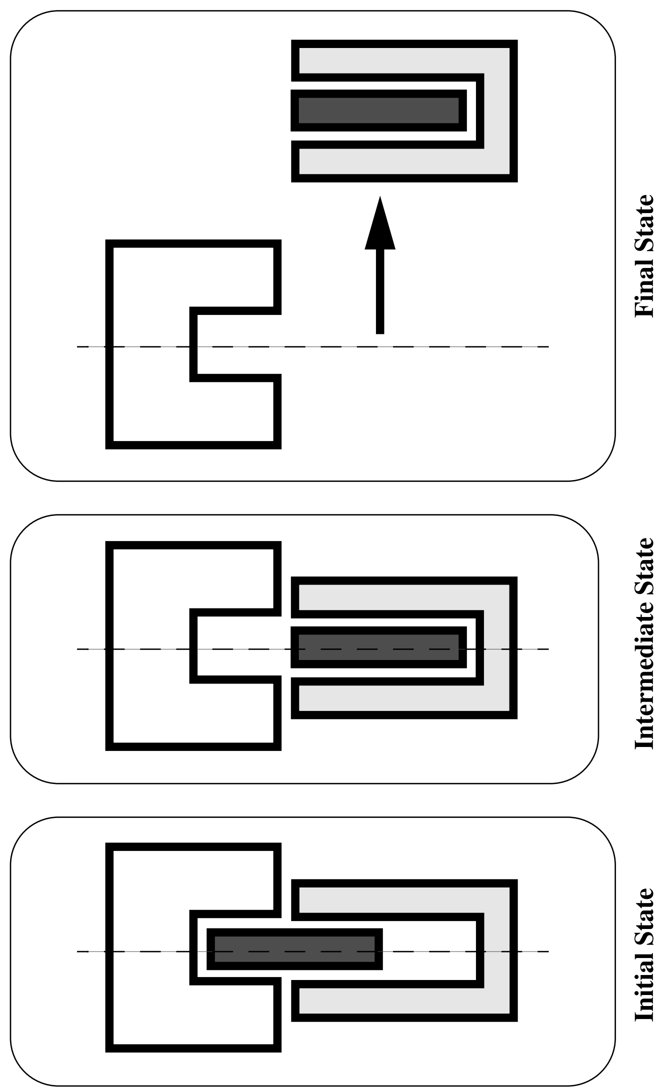

FIGURE 1b: The behavior of a door lock.

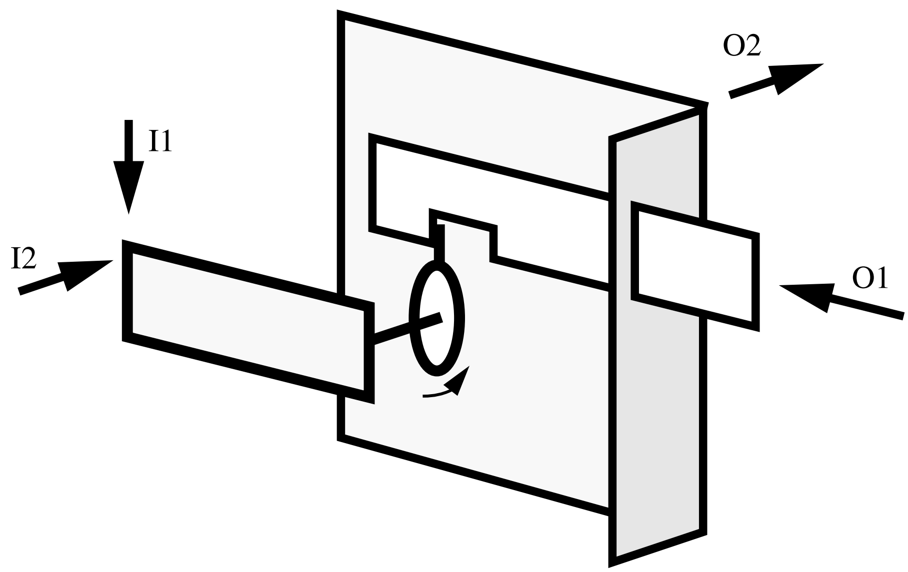

FIGURE 2a: A door lock using a lever and a cam.

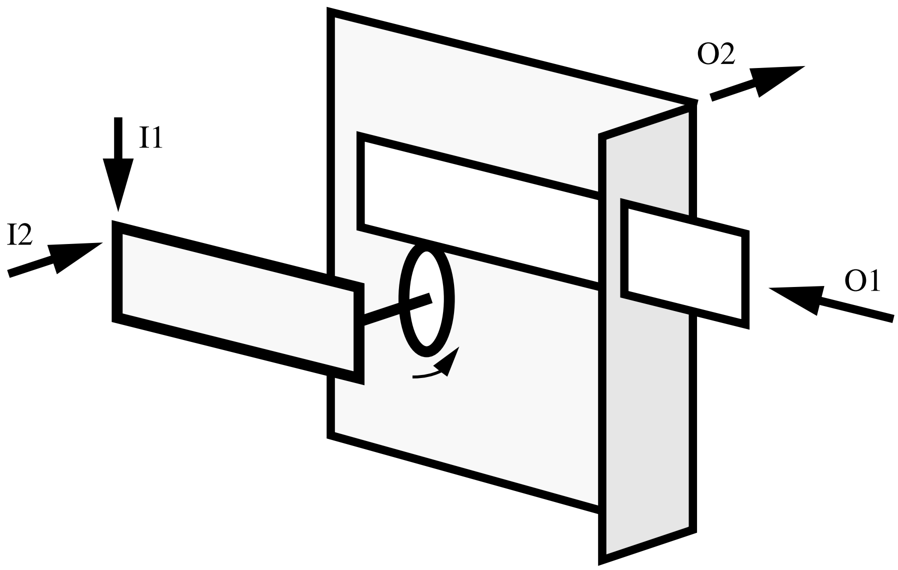

FIGURE 2b: A door lock using a lever and a roller.

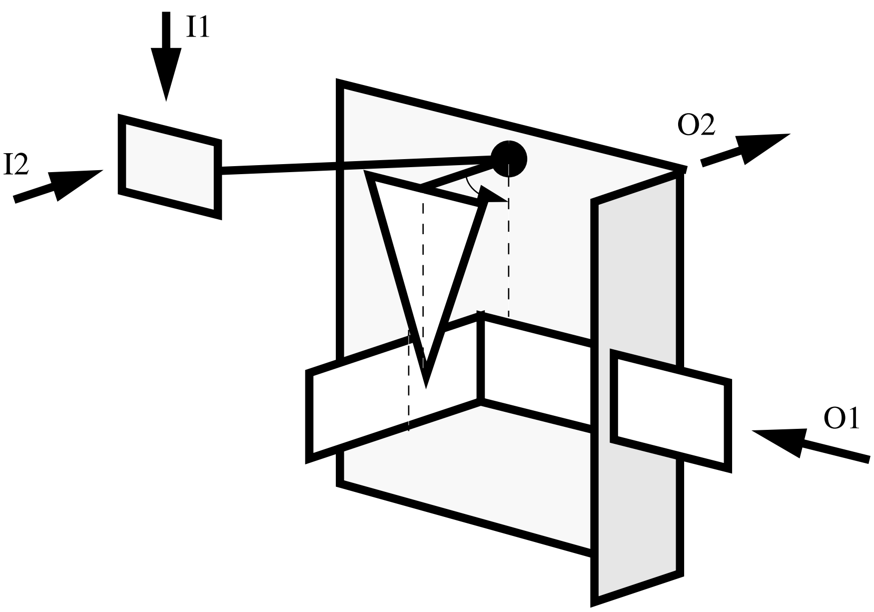

FIGURE 3: A door lock using two levers and a wedge.

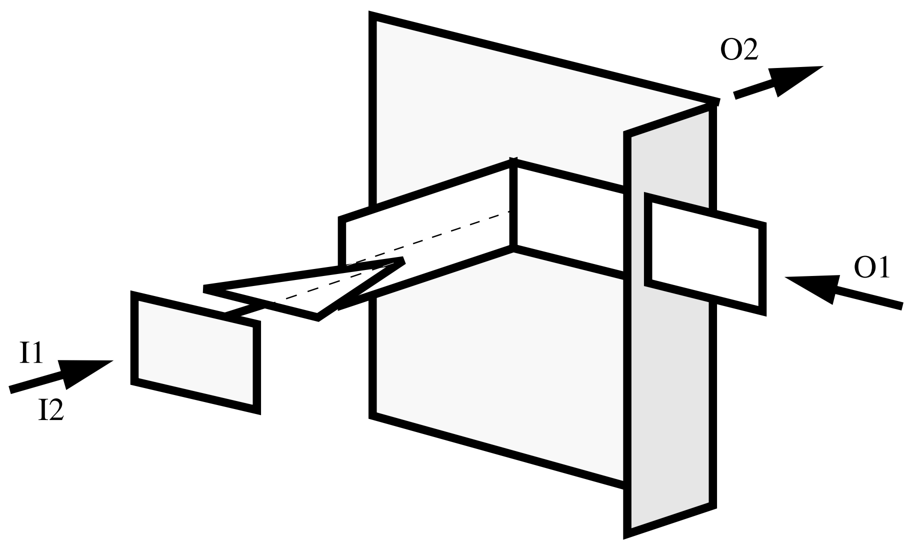

FIGURE 4: A functionally simple door lock.

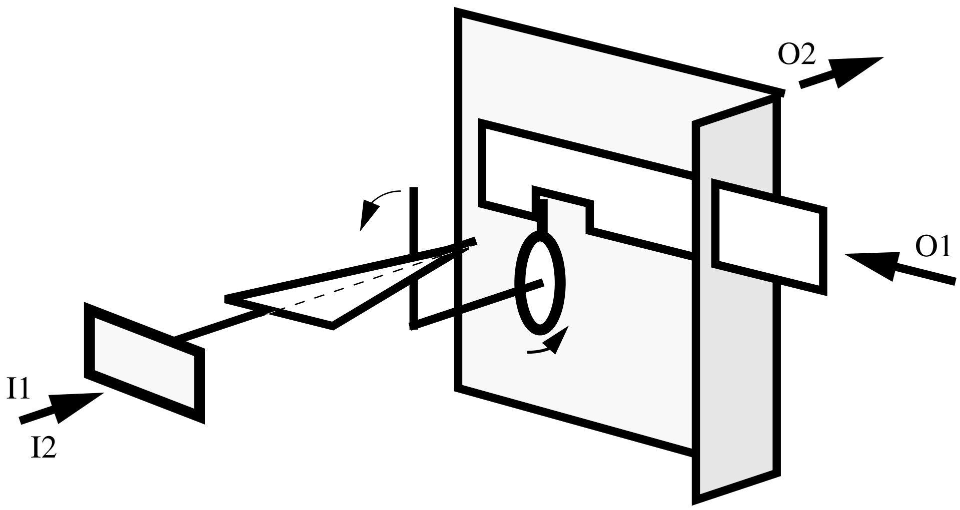

FIGURE 5: A functional simplification of the door lock in Figure 2b.

```
(design d1
        (name        "door lock with lever, roller and bolt")
        (type        door-lock)
        (part-of)
        (structure   (attributes(position (shut open)))
                     (componentsd2 d3 d4) (relations r1 r2)
                     (inputs (component d2))
                     (outputs(attribute position) (component d4 position)) )
        (behavior    (state  state1 (position shut) (component d4 (position out))
                     (state  state2 (position shut) (component d4 (position in)))
                     (state  state3 (position open) (component d4 (position in)))
                     (state-transition b0
                           (state1 state2
                             (causal(input (d2 down))
                               (behavior d2 b1)(behavior d3 b3)(behavior d4 b5)))
                     (state2 state3
                            (causal(input (d2 forward))) ) )
        (function    (input (d2 down) (d2 forward))
                     (output (state1) (state3)) (by (behavior b0)) )
)

(connected-relation r1(part-ofd1) (members (d2 rotor) (d3)) )
(touching-relation r2 (part-ofd1) (members (d3) (d4)) )

(design d2
        (name        "door handle")
        (type        lever)
        (part-of     d1)
        (structure   (attributes(arm (up down)) (rotor (0 angle)))
                     (components) (relations) )
        (behavior    (b1    (state (arm up)   (rotor 0))
                            (state (arm down) (rotor x)))
                     (b2    (state (arm down) (rotor x))
                            (state (arm up)   (rotor 0))) )
        (function    *** a default function should be initialized here
                     *** not currently implemented )
)

(design d3   ...etc...  )
(design d4   ...etc...  )
```

FIGURE 6: An edited partial representation of the design in Figure 2b.

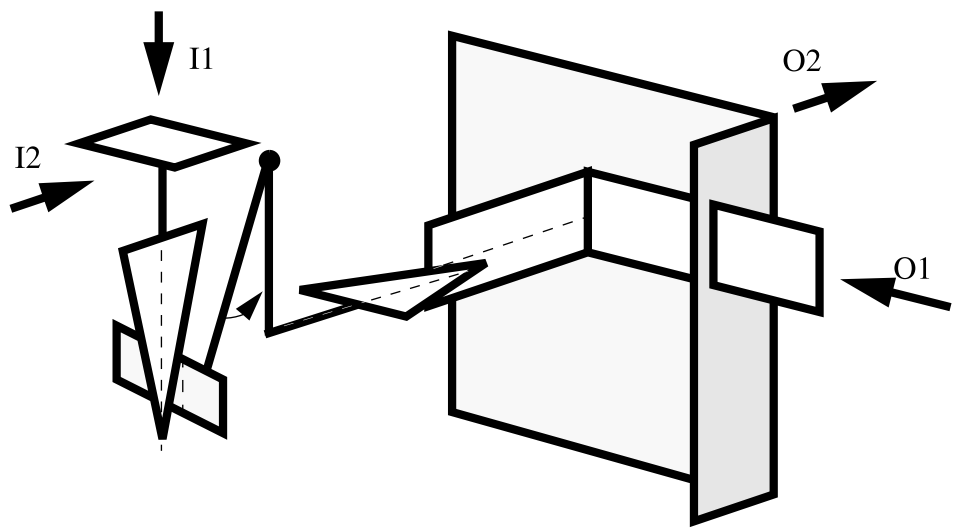

FIGURE 7: A door lock for structural and then behavioral simplification.