

11

Mutation and analogy to support creativity in computer-aided design

John S. Gero, Mary Lou Maher

Design Computing Unit
University of Sydney
NSW 2006 Australia

Creative processes in computer-aided design are identified as those that introduce new design variables into the design process. This paper describes the ideas behind the development of a computational model of creative processes in design. This computational model is founded on the use of mutation and analogy processes working in tandem using design prototypes as the representation schema.

Introduction

Creative design while highly valued is not currently supported by computer programs used during design. Research in design and artificial intelligence shows potential for improving our understanding of design and establishing computational models for creative design. Research in computational processes for design has been primarily concerned with routine design which has not provided much insight into the adequacy of these processes to support creative design. The distinction between routine and creative design is subjective but an operational definition can be given. Routine design is the result of making design decisions in the context of a design situation where all the decision variables are known a priori. Thus, in routine design the designer operates within a defined, closed state space of possible designs where the differences between designs can be characterised largely by the values selected for the design variables. To a lesser extent it can be further characterised by the selection of which variables to include in the design from a predefined range of variables. Creative design occurs when new design variables are introduced in the process of designing. Thus, in creative design the designer operates within a changing state space of possible designs; a state space which increases in size with the introduction of each new variable, Figure 1.

This paper describes the ideas behind the development of a computational model of creative processes, where creative processes in computer-aided design are identified as those that introduce new design variables into the design process thus providing the opportunity for novel designs to result. Two computational processes being explored are mutation and analogy working in tandem, where mutation involves the use of mutation operators to produce new variables and analogy finds similar situations and extracts knowledge from them to assist in the incorporation of the new variables into the current situation.

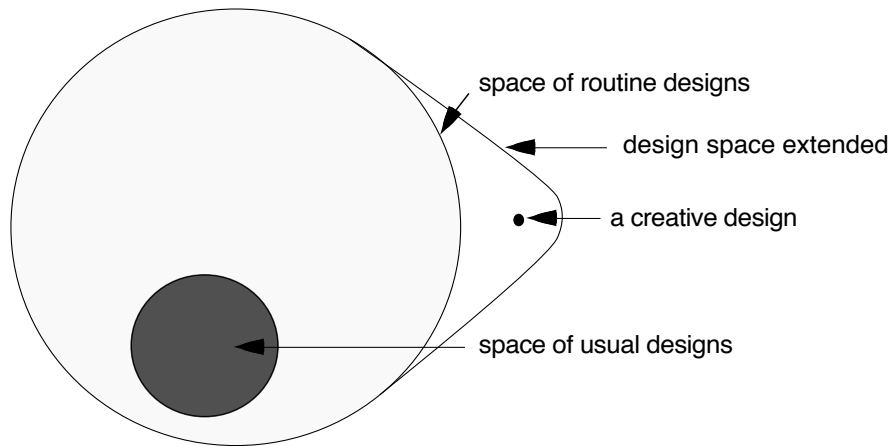


Figure 1. State space of routine and creative designs

Analogy and Mutation as Computational Processes

Analogy is a particularly useful approach for solving an unfamiliar new problem without adequate or directly applicable knowledge. Various forms of analogical reasoning in computer systems is an active research area in artificial intelligence and its application (Prieditis 1988). Based on the nature of the knowledge transferred from previous experience to the new problem, the analogical reasoning approach can be classified into two categories: transformational analogy and derivational analogy (Carbonell 1982, 1983, 1986). Transformational analogy adapts the solutions to the past problems for the new problem. Derivational analogy applies the past problem solving processes or methods to solve the new problem. By analogy, the relations between the new problem and some past experience or knowledge about a particular phenomenon can be found. This experience or knowledge can be placed in the new situation so that the new problem can be better understood or a new plan for solving the problem can be generated. An example of an implementation that uses analogy to improve the efficiency of the search for alternative designs is reported in (Zhao and Maher 1988). An example of a knowledge based system for creative mechanical design, EDISON (Dyer et al 1986), uses analogy as a mechanism for invention.

Mutation is the deliberate action of changing features or attributes of an object or a concept in an unconventional manner. By unconventional manner we mean that the change is not restricted by the common rules and constraints. The purpose of mutation is to produce new properties, functions and meanings of an old object or concept by changing some of its basic features. Mutation can be performed in many different ways. Some variations of mutation are multiplication, division, addition, subtraction, and substitution operations performed on the features, functions and mechanisms of concepts. Computerization of mutation has been successful when the representation was appropriate; AM (Lenat 1982, Lenat and Brown 1983) is an example of applying simple mutation in the domain of mathematics. In complex representations computers still do not have adequate knowledge and it is difficult to obtain meaningful mutation operations. This may often make mutation seem somewhat arbitrary. Without appropriate guidance, its nature is to some extent opportunistic and success can not be guaranteed.

Supporting Creativity in Computer-Aided Design

In order to make use of analogy and mutation to support creativity in computer-aided design, three issues have been identified:

- (i) the representation of design experience,

- (ii) the identification and application of mutation operators, and
- (iii) the development and application of analogical reasoning.

Representation of design experience

Design experience provides the source of knowledge for both mutation and analogy. For mutation, experience provides a typical design which forms the basis of mutation. For analogy, design experience provides various design situations from which an appropriate analogy can be drawn. To distinguish the representation of previous design experience from the current design situation, we introduce the terms *source* and *target*. Source knowledge is a representation of previous design experience in a particular domain, such as building design. This is in contrast to a target design space which provides a representation of the current design problem and its context.

The representation to be used for the source space is the schema *design prototypes* developed at the University of Sydney (Gero 1990). A design prototype is a generalization of groupings of elements in a design domain which provides the basis for the commencement and continuation of a design process. The design prototype thus represents generalizations of previous designs from which new design solutions applicable to the current situation can be derived. To facilitate design reasoning and support mutation and analogy, a design prototype generalizes previous designs according to the following categories:

- Functions are the design goals that can be achieved by using the design prototype. The functions of a design prototype are a primary consideration in determining whether it is suitable for a particular design context. In architectural design this may include, within a particular context, style, type of building, purpose and intent.
- *Structure* information describes the design in terms of its physical existence or the conditions for such existence. Structure is described in terms of decision variables whose values will be determined during the design process. In a building this may include the topology of the spaces, their geometry and the materials of construction of the bounding elements to those spaces.
- *Behaviours* are the expected responses of the design in the current design context. In architectural design, for example, some behaviour characteristics are efficient circulation, adequate lighting, thermal comfort, and cost.
- *Dependency knowledge* provides the relations between function, behaviour and structure. For example, the behavior cost may be related to the structure material.

This explicit representation of design knowledge can support creativity because it goes beyond the representation of only decision variables as is customary in mathematical models of design processes (Radford and Gero 1988). The mapping from function to structure represents available heuristics for producing design solutions using knowledge-based expert system approaches. The mapping from structure to behaviour is explicitly available through analytical processes. The dependency knowledge provides the basis for reasoning about the associations and potential values of related variables within a design context. Figure 2 illustrates a generalized dependency network where the progression from left to right represents the relations between functions (F), behaviours (B), behaviour variables (BV), structure variables (SV) and structures (S). An example of part of a dependency network for a design prototype representing generalized double hung windows is shown in Figure 3.

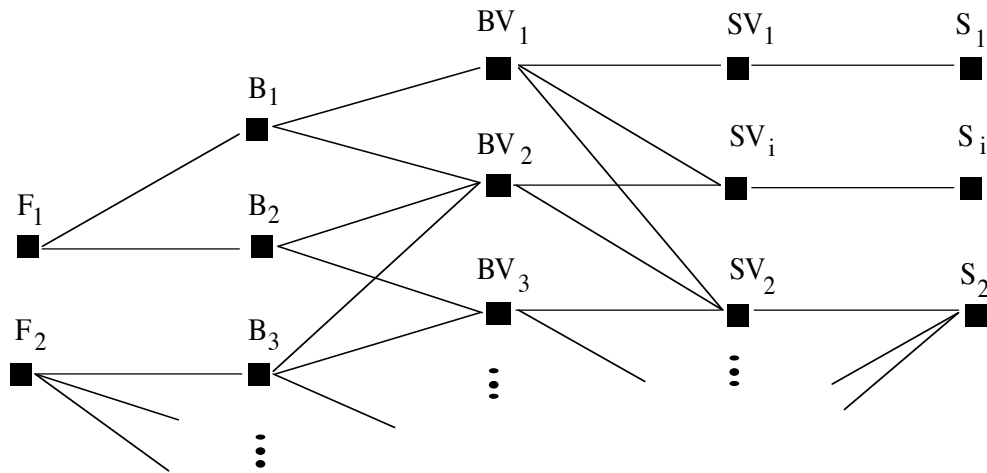


Figure 2. A dependency network in a design prototype showing the relationship among function, behaviour, behaviour variables, structure variables and structure

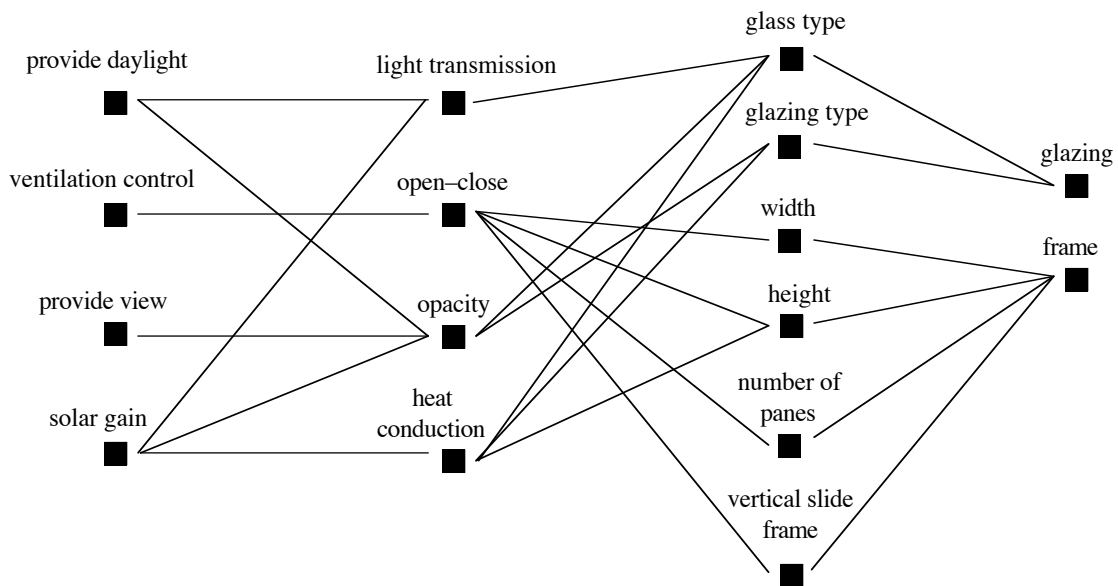


Figure 3. Part of the dependency network for the design prototype concerned with double hung windows

Mutation operators for creative design

Mutation operators are used to introduce new decision variables in the target space. Such mutation operators might be multiplication, where a decision variable becomes two variables of the same nature; or addition, where a decision variable of a different nature is introduced.

Mutation operators are applied to structure decision variables and are used to introduce one or more new variables into a design. There are two classes of mutation operators:

- (i) homogeneous, i.e., those which produce new variables of the same type as the variable being mutated, and
- (ii) heterogeneous, i.e. those which produce new variables of a different type.

Homogeneous operators include division where a single variable, X, is divided into two variables, X1 and X2, thus introducing a second variable of the same type as X. As a result new structures can be produced. The issue of homogeneity is important since homogeneous variables can use the existing dependency information in the design prototype from its precursor variable, Figure 4. Other homogeneous operators include multiplication and exponentiation, although only division has been reported in Cagan (1990). In the dependency network of the window prototype, a homogeneous mutation could be the division of the structure variable height into height1 and height2. The relations in the dependency network are modified as shown in Figure 5.

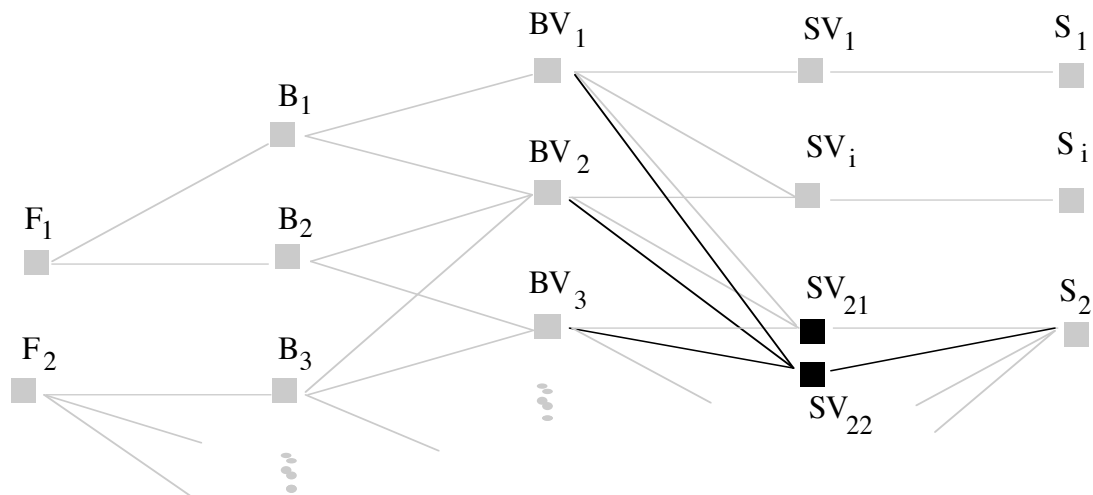


Figure 4. Modified dependency network after a homogeneous mutation operation on SV2 where SV2 has been divided into {SV21, SV22}.

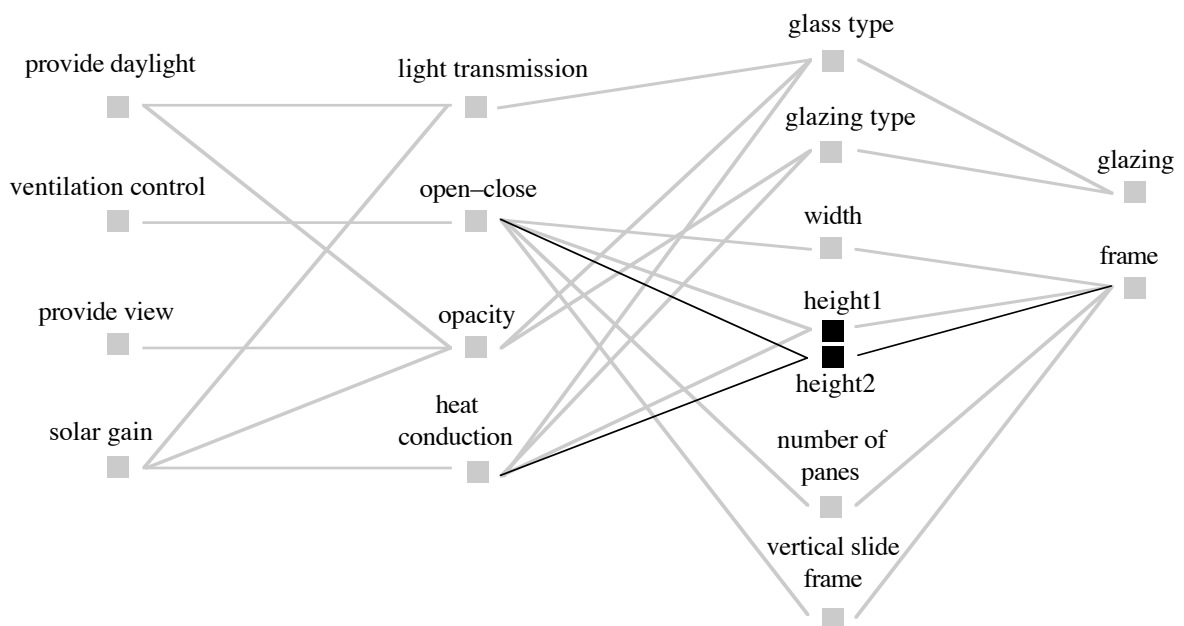


Figure 5. Modified dependency network for the double hung window design prototype after a homogeneous mutation operation

Heterogeneous operators include addition and the set operations of union and intersection in general. An example of using addition as a mutation operator is to select the structure variable length and add as a new variable the variable angle to it. This results in a length that can be rotated where previously it was fixed in direction. Heterogeneous variables, however, require additional dependency information for their inclusion in the design prototype since no such information currently exists in the design prototype, Figure 6. In the window design prototype, the use of an addition mutation operator to produce the rotation variable is illustrated in Figure 7.

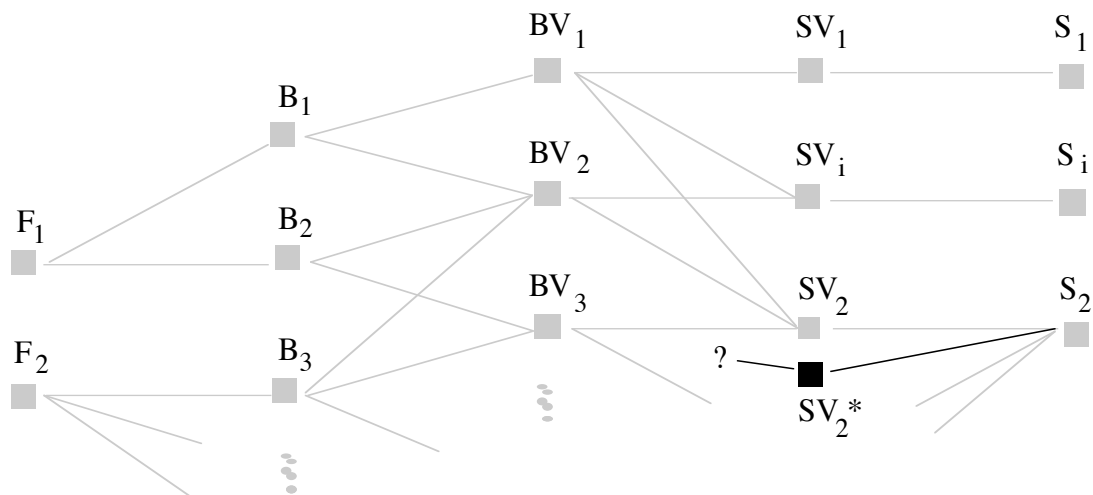


Figure 6. In a heterogeneous mutation a new variable SV_2^* is introduced through an addition mutation process and has no behaviour dependency knowledge available

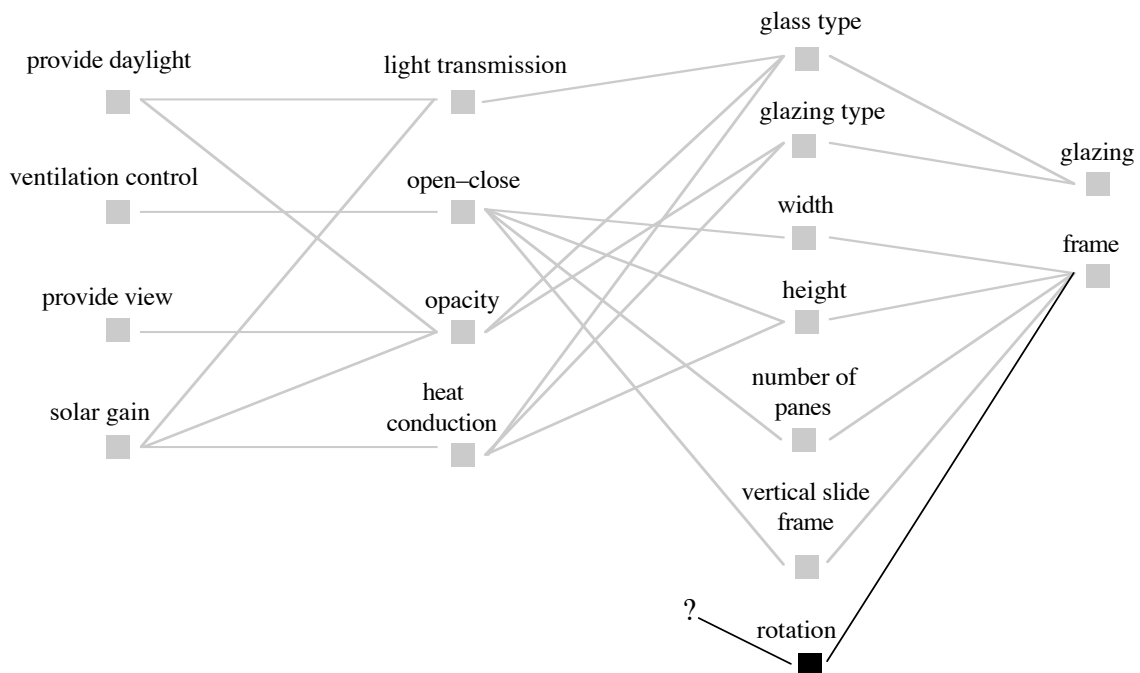


Figure 7. The dependency network of the double hung window design prototype after the addition of the rotation variable

Analogical reasoning tasks are identified for finding relevant knowledge in the source space to accommodate a new heterogeneous design variable in the target space. Once a new variable has been introduced the issue of how to integrate it into the current situation arises. Here we use existing techniques and approaches for analogical reasoning to identify relevant design prototypes which potentially contain knowledge appropriate for the integration of that new variable. Analogical reasoning can be considered in two major steps: finding an analogous situation and using the knowledge from the previous situation in the new situation. The techniques derived from research in artificial intelligence that are relevant to supporting analogical reasoning in creative design processes are memory indexing and transformation.

Memory indexing. Memory organization is important when attempting to make associations between design prototypes that may be very different except for a few shared attributes. Indexing is used to facilitate information flow across contexts, on which analogical reasoning is based. Design prototypes are grouped and retrieved according to their similarities of characteristics through indices. The indices are related to specific manifestations of the design prototype knowledge categories, i.e. functions, structures and behaviours. To support the mutation of a prototype, the dependency networks of the source prototypes are used. If a structure variable is introduced, the dependency networks are indexed by their structure variables. For example, the introduction of the variable, rotation, in the double hung window requires finding previous experience in which rotation is a structure variable. The search for relevant design prototypes in this example may result in the retrieval of a revolving door design prototype. Figure 8 illustrates the dependency network of a revolving door design prototype.

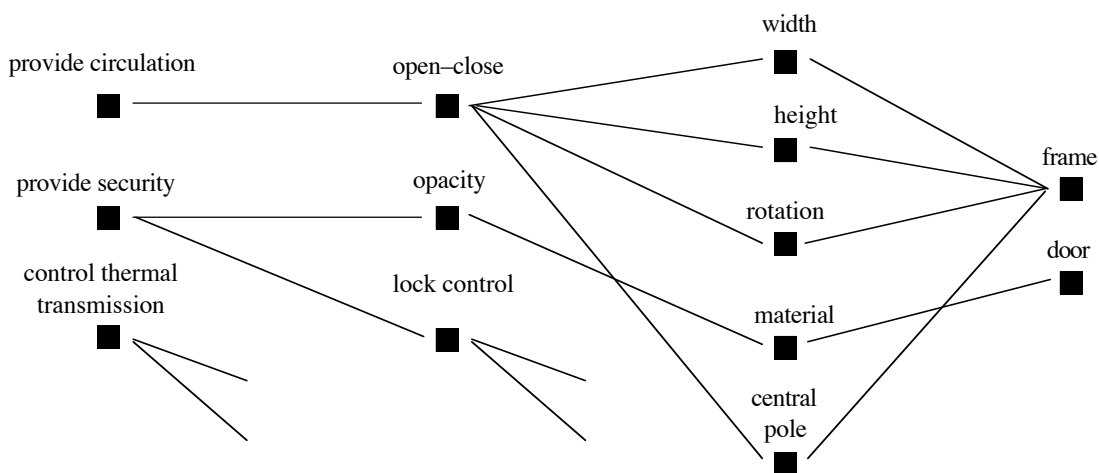


Figure 8. The dependency network of the revolving door design prototype

Transformation. Once a potentially appropriate source design prototype is identified, a transformation of that design prototype's knowledge is needed to establish its relevance to the new design context. In the model for creative design being used here, the current context is defined by the selected design prototype in the target space, for example, the double hung window. The source knowledge for accommodating the introduction of a new variable in the target space is the retrieved design prototype, for example, the revolving door. The transformation from the source space to the target space requires selecting the relevant parts of the dependency network and combining this with the dependency network in the target space, Figure 9.

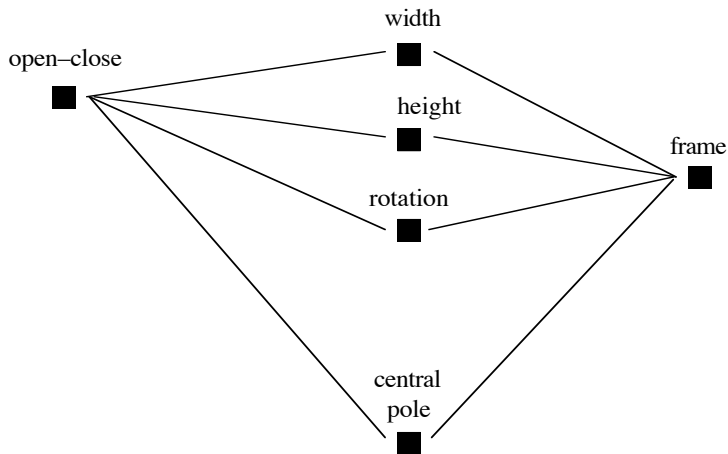


Figure 9. Part of the dependency network of the revolving door design prototype retrieved in the search for another design prototype containing the structure variable rotation

This can be executed using the structure-mapping approach of Falkenhainer et al (1989/90) where the indexed structure variable acts as a focus and the related structure and behaviour variables are mapped from the source design prototype onto the target design prototype. Clearly, there must be a map to part of the structure otherwise the new variable is not part of the structure. However, new behaviour variables may result which are part of existing behaviours. Alternately, new behaviour variables may introduce new behaviours which may or may not be part of existing functions. If they are not a part of existing functions then a new function is introduced. For the window example, the combined dependency network is illustrated in Figure 10. This 'new' window operates in manner similar to that of a revolving door and is reflected in the dependency network by the introduction of the new nodes 'rotation' and 'central post'.

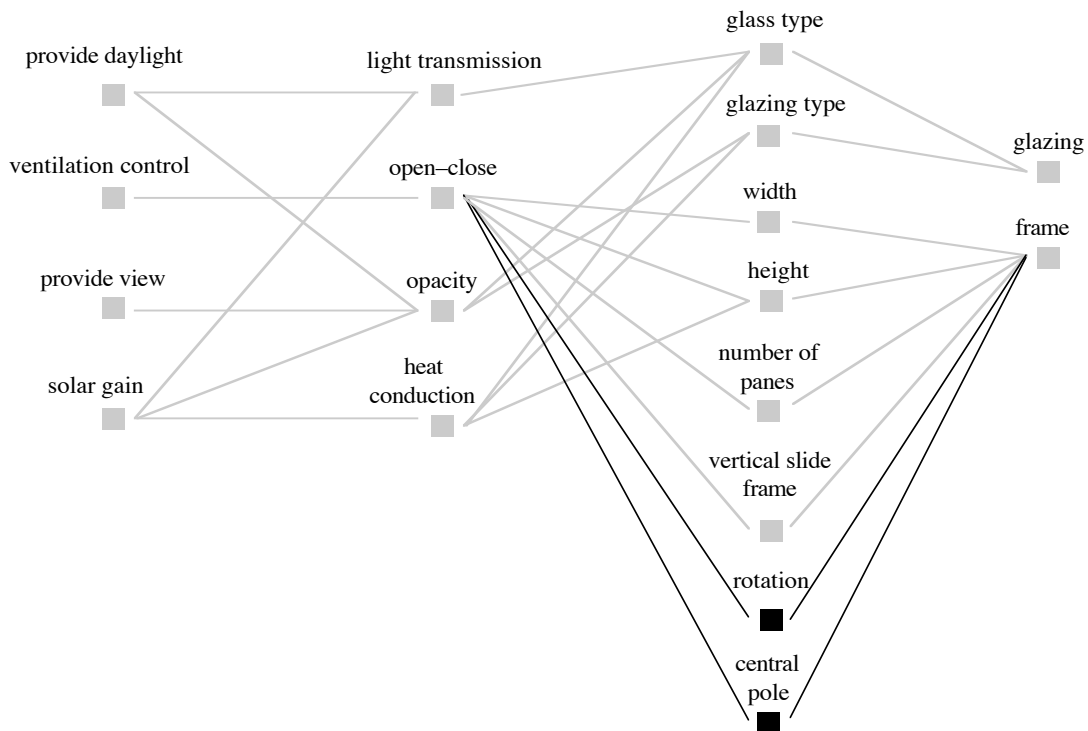


Figure 10. Combining the relevant part of the dependency network of the revolving door design prototype with the existing window design prototype, where the bold relations and nodes represent the knowledge imported from the revolving door to create a new window design.

Discussion

The ideas in this paper have been partially implemented. The dependency networks and associated design prototypes have been implemented to support routine design at the University of Sydney and for creative design at Carnegie Mellon University. The implementations have not progressed as far as the ideas described here. As the implementations proceed, the need for more knowledge in the dependency networks and the implications of the application of mutation operators on the dependency networks becomes more explicit.

The computational processes of mutation and analogy provide approaches to supporting creative design but their application is strongly dependent on the representation of the source and target spaces. In our model we use a representation centred on the concept of design prototypes because they provide a rich environment for organising design knowledge. The explicit representation of function, behaviour, and structure as attributes and dependencies facilitates mutation and analogical reasoning. A different representation of design knowledge, such as logic, rules, or design grammars, may invalidate much of the effectiveness of the application of mutation and analogy as described in this paper.

Acknowledgements. The authors wish to thank Fang Zhao, Jun Jo and Lena Qian with whom they have had fruitful discussions.

References

- Cagan, J. 1990. *Innovative Design of Mechanical Structures*. PhD Thesis (unpublished), Department of Mechanical Engineering, University of California, Berkeley.
- Carbonell, J. G. 1982. "Towards a computational model of metaphor in common sense reasoning." *Proc. Fourth Annual Meeting of the Cognitive Science Society*.
- Carbonell, J. G. 1983. "Learning by analogy: formulating and generalising plans from past experience." In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (eds), *Machine Learning: An Artificial Intelligence Approach*. Palo Alto, California: Tioga.
- Carbonell, J. G. 1986. "Derivational analogy: a theory of reconstructive problem solving and expertise acquisition." In R. S. Michalski, J. G. Carbonell and T. M. Mitchell (eds), *Machine Learning II: An Artificial Intelligence Approach*, Los Altos, California: Morgan Kaufmann.
- Dyer, M. G., Flowers, M. and Hodges, J. 1986. "EDISON: an engineering design invention system operating naively." In D. Sriram and R. Adey (eds), *Applications of Artificial Intelligence to Engineering Problems*, Berlin: Springer-Verlag.
- Falkenhainer, B., Forbus, K. D. and Gentner, D. 1989/90. "The structure-mapping engine: algorithm and examples." *Artificial Intelligence* 4. no.1: 1-63.
- Gero, J. S. 1990. "Design prototypes: a knowledge representation schema for design." *AI Magazine* 11, no.4: 26-36.
- Lenat, D. B. 1982. "An artificial Intelligence approach to discovery in mathematics as heuristic search." In R. Davis and D. B. Lenat (eds), *Knowledge-Based Systems in Artificial Intelligence*. New York: McGraw-Hill.
- Lenat, D. and Brown, J. S. 1983. "Why AM and Eurisko appear to work." *Proc. AAAI-83*, pp. 236-240.
- Prieditis, A. 1988. *Analogica*. London: Pitman/Morgan Kaufmann.
- Radford, A. D. and Gero, J. S. 1988. *Design by Optimization in Architecture and Building*. New York: Van Nostrand Reinhold.
- Zhao, F. and Maher, M. L. 1988. "Using analogical reasoning to design buildings." *Engineering with Computers* 4: 107-119.

This is a copy of the paper: Gero, J. S. and Maher, M. L. (1991). Mutation and analogy to support creativity in computer-aided design, in G. N. Schmitt (ed.), *CAAD Futures '91*, ETH, Zurich, pp. 241-249.