Using conceptual clustering to learn about function, structure and behaviour in design

Leila Alem, Mary Lou Maher

INRIA

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1)39635511

Rapports de Recherche

N° 1401

Programme 5
Traitement du Signal,
Automatique et Productique

USING CONCEPTUAL CLUSTERING TO LEARN ABOUT FUNCTION, STRUCTURE AND BEHAVIOUR IN DESIGN

Leïla ALEM
Mary Lou MAHER

Mars 1991

RR-1401

Utilisation du clustering conceptuel pour l'apprentissage automatique des notions de fonction, structure et comportement en conception

Leïla Alem et Mary Lou Maher

Departement of Architecture and
Design Science University of Sydney
NSW 2006 Australia

Résumé

La modélisation des connaissances en conception est un sujet de recherche de pointe abordé par les ingénieurs, les architectes et les chercheurs en intelligence artificielle pour le développement de base de connaissances en conception. Nos principaux pôles d'intérêts sont la représentation des connaissances et l'acquisition des connaissances. La construction de bases de connaissances en conception est difficile car l'homme acquiert la connaissance en conception essentiellement au travers de son expérience se traduisant par une connaissance diverse et non structurée. Parmi les branches de l'apprentissage automatique, le " clustering" conceptuel offre des techniques pour structurer des observations en concepts généralisés. En particulier, une de ces techniques appelée "formation de concept" s'avère prometteuse pour l'apprentissage de concepts en conception. Une programme de clustering conceptuel permet d'apprendre la connaissance en conception en traitant séparément les notions de fonction, structure et comportement. Les clusters ainsi obtenus servent de base pour le calcul des associations entre fonction, structure et comportement. On obtient ainsi une représentation des connaissances en conception qui peut être utilisée pour assister le concepteur au cours de la phase de synthèse du processus de conception.

Using Conceptual Clustering to Learn about Function, Structure and Behavior in Design

LEILA ALEM and MARY LOU MAHER

Department of Architectural and Design Science
University of Sydney, NSW 2006 Australia
e-mail: mary@archsci.su.oz.au
fax: 0011-61-2-692-3031

Abstract

Modeling design knowledge is an active research topic among engineers, architects, and AI researchers interested in knowledge-based design. The primary concerns are knowledge representation and knowledge acquisition. Knowledge acquisition for knowledge-based design is difficult because humans acquire design knowledge primarily through experience resulting in a diverse and unstructured body of knowledge. Conceptual clustering, an area of the broad field of machine learning, provides techniques for structuring observations into generalized concepts. Specifically, concept formation shows promise for learning design concepts. A conceptual clustering program is used to learn about design by clustering function, structure, and behavior. These clusters are then used as the basis for learning associations between function, structure and behavior. The resulting representation of design knowledge can be used to provide assistance during the synthesis stage of the design process.

1. Introduction

Knowledge-based systems for design can be broadly classified into generative and interpretive systems. Generative systems are concerned with producing a design solution or solutions from a set of specifications. Interpretive systems are concerned with producing an evaluation from a description of a design solution. The type of knowledge needed for a generative system differs from that needed for an interpretive system. Interpreting a design solution to produce an evaluation, e.g. whether it satisfes regulatory and/or safety contraints, requires analytical knowledge. Producing a design solution, e.g. finding an appropriate description, requires synthetic knowledge. Although both types of design knowledge include hueristics as well as domain theory, the application and representation of the knowledge varies. In this paper we are concerned with the representation and acquisition of synthetic knowledge, more specifically, design knowledge required to generate one or more solutions for a set of design goals and requirements.

Many knowledge-based systems for design synthesis have been developed. The early examples

used a rule-based representation of design knowledge. For example, R1 (McDermott 1980) is an expert system for configuring computer systems implemented in OPS5 (Forgy 1981) and VEXED (Mitchell et. al. 1984) is an expert system for VLSI design in which design knowledge is represented primarily as refinement rules. Other knowledge-based systems for design synthesis use a combination of object-centered and rule-based representations. For example, HI-RISE (Maher 1984) designs structural systems for high rise buildings in which the design knowledge is represented in both frames and rules, DSPL (Brown 1985) is a language for design knowledge that represents knowledge in a hybrid form using agents, constraints, etc. The difficulty in developing and maintaining knowledge-based design systems lies in acquiring the design knowledge in a form that is suitable for representation in a knowledge base. The most common approach to building a knowledge base is to formalize generalized knowledge in a particular domain and encode the generalizations in a programming or knowledge representation language.

Generalizing design knowledge is particularly difficult because expert designers acquire and use their knowledge through experience. There is very little design synthesis knowledge in text books or taught in school. The result of this is that designers find it difficult to articulate their knowledge and tend to describe their knowledge through examples of design situations. This lack of a coherent body of generalized design knowledge has lead to difficulties in developing design knowledge bases that are acceptable or useful to designers.

Machine learning can be considered to comprise four learning paradigms (Carbonell 1990): the inductive paradigm, the analytic paradigm, the genetic paradigm, and the connectionist paradigm. The inductive paradigm is of interest here because it provides a set of techniques for automatically developing generlized concepts from examples of problems and solutions. These techniques are of interest in developing design knowledge bases since the use of examples to describe design knowledge is practiced by designers more readily than producing a generalized body of design knowledge.

The use of inductive learning in knowledge-based design requires more than just applying a technique to a domain. The results of design research in developing knowledge-based design systems provides some insight into the nature of the design knowledge that needs to be included in the knowledge base. This background can assist in selecting and adapting an inductive learning technique that can be used to learn knowledge about design synthesis.

In this paper we present conceptual clustering as an inductive learning technique that is relevant to learning design knowledge. Then we discuss the representation of design knowledge that facilitates the synthesis of design solutions. In the following sections we describe our approach to combining the ideas of conceptual clustering with design knowledge representation to learn design prototypes. We conclude with a discussion of the implementation of these ideas and directions for further work.

2. Inductive Learning

Inductive learning is a subset of machine learning in which the learning program accepts a set of instances and produces a set of concepts that are generalizations of the instances. Inductive learning can be categorized according to the type of input to the learning program. Two distinct types of input are: examples and observations. Examples are instances that are classified by the teacher as positive or negative instances. Observations are instances that are not classified by the teacher.

Two characteristics of inductive learning are incremental and empirical learning. Incremental learning indicates that the learning program is able to modify its generalized representation incrementally, as new instances are introduced. Although the early inductive learning programs were not incremental, the majority of the effort in developing inductive programs currently provide for incremental learning. Empirical learing indicates that domain knowledge is not used to develop generalizations from instances. Inductive learning relies entirely on observations or examples to produce a set of concepts.

One of the most widely studied areas of inductive learning is conceptual clustering (Michalski and Kodratoff 1990), also called concept formation. In conceptual clustering, classes of observations are created using a numerical measure of similarity. Classes (we will refer to these classes as clusters) are collections of observations whose intra-class similarity is high, that is the observations in the cluster are similar, and inter-class similarity is low, that is the observations in two different clusters are not similar. Intra-class similarity is represented by $P(A_i = V_{ij} / C_k)$ and inter-class by $P(C_k / A_i = V_{ij})$ where $A_i = V_{ij}$ is an attribut-value pair and C_k is a class or cluster. A cluster represents a class, but unlike an object-oriented definition of a class, a cluster is induced from the examples and explicitly contains the examples of the class.

Examples of programs that use the conceptual clustering approach to inductive learning include EPAM (Feigenbaum 1984), UNIMEM (Lebowitz 1987), and COBWEB (Fisher 1987). Although the details differ from program to program, a conceptual clustering program accepts a set of observations as input and produces a hierarchy of clusters as output. An observation is described by a set of attribute-value pairs. A cluster is represented by a subset of the attribute-values pairs. Each cluster is also defined by the observations stored below it, and contains the observations of all it sub-clusters. When a new observation is introduced, the new observation is accommodated in the existing hierarchy using an evaluation function to determine the most appropriate cluster or to introduce a new cluster.

In COBWEB, the probability associated with each cluster and with each attribute-value pair is stored; the resulting representation is called probabilistic concept. COBWEB performs one of several operators to accomodate a new observation, and uses a category utility function to select between the availables operators.The operators include:

- classifying the object with respect to an existing cluster,

- creating a new cluster,
- combining two clusters into a single cluster (merging), and
- dividing a cluster into several clusters (splitting).

The use of merging and splitting operators desensitize the hierarchy to the effects of input order.

The category utility, CU, is defined by Gluck and Corter (Gluck and Corter 1985) as "the increase in the expected number of property values that can be correctly guessed, given a set of categories (partition), over the expected number of correct guesses with no such knowledge." The CU is calculated thus

$$\frac{\Sigma_{k=1}^{n} P(C_k)[\ \Sigma_i \Sigma_j P(A_i = V_{ij} / C_k)^2 - \Sigma_i \Sigma_j P(A_i = V_{ij})^2\ ]}{n}$$

where n is the number of categories.

The resulting hierarchy is a probabilistic concept tree where the classification is done using a path of "best" matching nodes which depend on an object's attribute-value pairs. In contrast, in a classical hierarchy (decision tree) the decision is based on the value of a single attribute (Fisher 87). COBWEB is restricted to lists of nominal attribut-value pairs. Such a restriction is not appropriate for representing many real word domains (Reich 1990).

BRIDGER (Reich 1990) is an adaptation of COBWEB for engineering design applications. Bridger allows multiple attribute types such as: ordered, continuous, and partially ordered. The use of multiple types of attributes introduces shallow knowledge about attributes. BRIDGER maintains new features in addittion to the original attributes and can be viewed as an approach to constructive induction (Reich 1990) . In order to use the cluster hierarchy for design synthesis, BRIDGER predicts missing properties by assigning partial description characteristics describing the nodes traversed. Characteristics are attribute values that satisfy :

$P(A_i = V_{ij} / C_k) \geq$ threshold and

$P(C_k / A_i = V_{ij}) \geq$ threshold.

The threshold is usually fixed to 0.75.

3. Design Knowledge Representation

The implementation of design knowledge bases has been largely influenced by the available representation paradigms and the nature of the design process. As discussed previously, we are interested here in the representation of knowledge for the synthesis of design solutions. The representation of design knowledge can either be organized by the representation language being used or by the generalization of the design process and its associated knowledge requirements. Using a representation language as a guide for design knowledge representation results in a knowledge base that is stated in terms of rules, frames, objects, logic, etc. Although many design knowledge bases use these representations as a basis, such languages do not provide guidance in the generalizations relevant to the task of designing.

More recent efforts in developing knowledge-based design systems have identified appropriate process models for design and representation paradigms specific to design synthesis. For example, Chandrasekaran describes a task oriented approach to representing design knowledge (Cnadrasekaran 1990), Maher describes various process models for design and their associated representation requirements (Maher 1990). In many cases knowledge bases for design are handcrafted by the knowledge engineer, usually resulting in generalizations that are made explicit to facilitate further knowledge base development. Two examples of this are the development of R1 followed by the development of SALT to faciliatate knowledge acquisition (Marcus 1988) and the development of HI-RISE followed by the development of EDESYN which captured the generalized representations for synthesis used in HI-RISE without the specific knowledge about building design (Maher 1988).

In this paper we adopt the view that design knowledge should be structured around an understanding of design rather than around a particular general purpose knowledge representation language. More specifically we adopt the concept of design prototypes as a useful representation paradigm for design knowledge. Design prototypes are introduced by Gero (Gero 1990), their implementation and application to design are described in (Gero et. al. 1988; Tham et. al. 1988).

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. A prototype represents a class of elements from which instances of elements can be derived. It comprises the knowledge needed for reasoning about the prototype's use as well as about how to produce instances in a given design context. A prototype can also be related to others either as a specialization or generalization or as a component or system to which other prototypes are the components. A hierarchy of prototypes can therefore be constructed. A designer's design knowledge may be considered as being comprised of a set of prototypes in his particular domain. Design using prototypes is a process in which suitable prototypes are sought for based on the given design specifications and are instantiated to produce instances that satisfy design goals and constraints.

In order to support design synthesis, prototypes provide generalizations for design knowledge that include function, structure, and behavior. Purely syntactic design knowledge, e.g. what a particular design looks like, is not sufficient for reasoning about design experience. To facilitate reasoning about a prototype's semantics as well as syntax in design, a prototype explicitly represents function, behavior and structure.

Functions are the design goals or requirements that can be achieved by using the prototype.

Structure attributes describe the prototype in terms of its physical existence or the conditions for such existence. These are typically design variables whose values will be determined during the instantiation process.

Behaviors are the expected reactions or responses of an instance of the prototype under the

possible design environment. Performance attributes of the prototype are the behaviors of particular interest in evaluating the appropriateness and ``goodness" of an instance of the prototype.

In addition to representing function, structure and behavior explicitly, design prototypes include the representation of associations between these knowledge categories. The associations that are useful for design synthesis are:

Function --> *Structure*: these associations are primarily the hueristics accummulated through design experience since there is no apparent function in structure or predetermined structure in function; during the early stages of design, the required functions may be known and the resulting structure is to be produced.

Function --> *Behavior*: a designer may use the associations from function to behavior to provide an intermediate statement of design requirements before structure is decided.

Behavior --> *Structure*: the associations between behavior and structure are the result of accumulated experience and analytical knowledge, providing a designer with options in generating structures that satisfy a set of behavior requirements.

4. Clustering and Associating Function, Structure and Behavior

Combining the available techniques in inductive learning and the use of prototypes as the basis for representing design knowledge as function, structure and behavior, we present a methodology for learning design knowledge from observations of design situations. The methodology draws primarily on the conceptual clustering approach to machine learning. However, conceptual clustering does not accommodate the categorization of attribute-value pairs as function, structure or bahavior. The methodology we propose goes beyond the conceptual clustering approach to include the identification of associations betweeen categories of clusters. The clusters and their associations are used to build a generalized class of design observations, similar to the concept of design prototypes.

The methodology has two distinct stages: (1) generating clusters of design knowledge and (2) finding useful associations between these clusters to identify design prototypes. The clusters are generated by considering observations whose attribute-value pairs are categorized according to function (F), structure (S), or behavior (B). The associations between clusters that are useful for design synthesis are:

F --> S,
F --> B, and
B --> S.

The implementation of this methodology is a program called DKAO. DKAO is implemented in CommonLisp using the frame-based representation language Framekit (Nyberg 1988). A portion of DKAO uses the BRIDGER program directly. The input to DKAO is a set of design situations (we will now refer to the design situations as observations to be consistent with the terminology introduced in Section 2) and the output is a set of design prototypes. The DKAO

program will be described in two parts:

Stage 1, where the observations are clustered according to function, structure, and behavior; and

Stage 2, where the clusters are grouped and association between function, structure and behavior groups are determined to identify design prototypes.

4.1. Generating clusters

Conceptual clustering algorithms provide a set of techniques that accept observations as input and produce clusters as output. The particular program used here is BRIDGER, an adaptation of COBWEB for design applications. The observations, or design situations, are described by a set of attribute-value pairs that are categorized according to function, structure, or behavior, as illustrated in Figure 1. An example of an observation for a bridge design is shown in Figure 2. The clusters that are the output of BRIDGER are therefore categorized according to function, structure, or behavior.

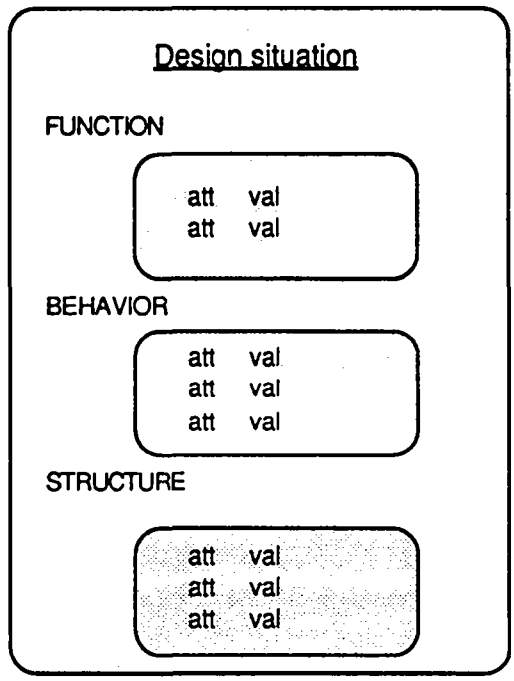


Figure 1: Representation of design situation or observation

The clustering process is illustrated in Figure 3. Each observation is characterized by three sets of attribute-values pairs; one set for each of function, structure and behavior attributes. These observations are transformed into three sets, where each set represents all observations of function, structure or behavior attributes. Each set is then input to the BRIDGER program to produce a hierarchy of clusters.

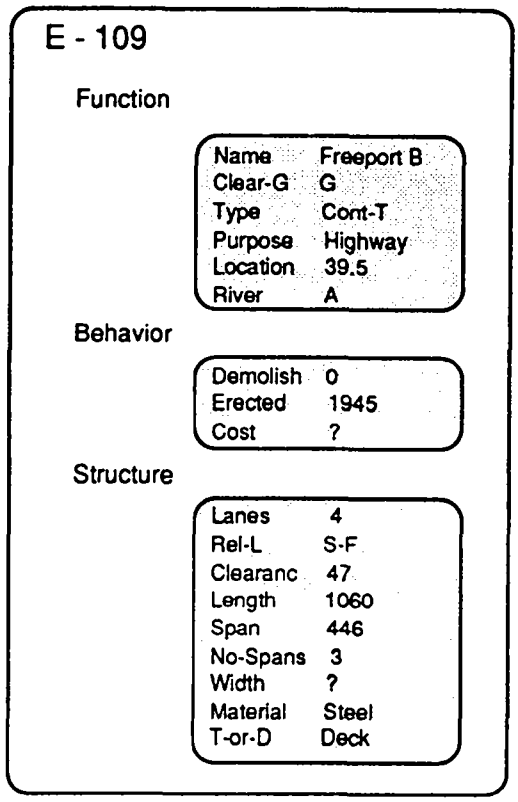


Figure 2: Example of a bridge observation

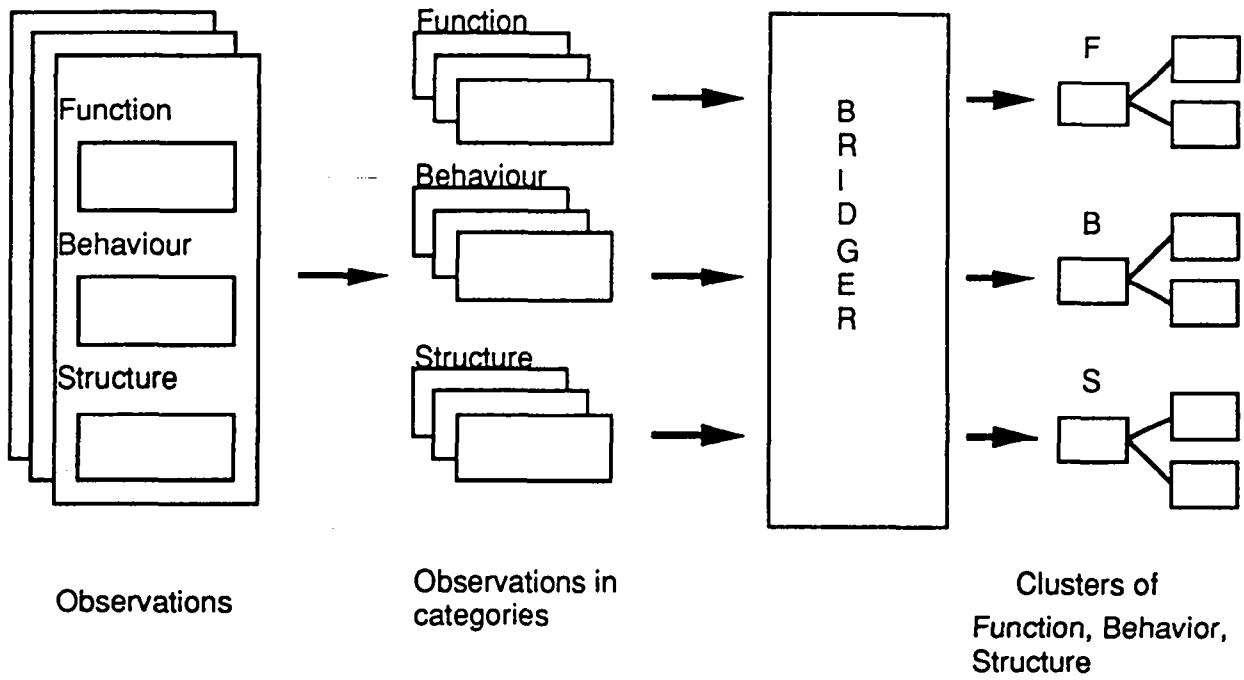


Figure 3: Clustering process

A cluster comprises a set of attribute-value pairs that is supported by one or more observations. An example of a cluster of structure attribute-value pairs for the bridge observations is shown in Figure 4.

S-G18
Parent : S-G14
Observations: 2
Atttribute SPAN, total times seen 2
value : LONG - 1 times
value : MEDIUM - 1 times
Attribute REL-L, total times seen 2
value : F - 2 times
Attribute MATERIAL, total times seen 2
value : STEEL - 2 times
Attribute LENGTH, total times seen 2
value : (910.0 3200.0 2) - 2 times
Attribute LANES, total times seen 2
value : (6.0 0.0 2) - 2 times
Attribute T-OR-D, total times seen 2
value : THROUGH - 2 times
Sons : E90-S
E84-S

Figure 4 : An example of structure cluster for bridge observations

4.2. Detemining associations for design prototypes

From the hierarchy of clusters produced by BRIDGER, only those clusters that are supported by more than one example are selected to be groups of attributes. The implication is that generalizations of design knowledge are based on more than one observation. In Figure 5 a representation of five clusters produced by BRIDGER is shown as a hierarchy, the relevant groups in this hierarchy are CLUST-12, CLUST-13, and CLUST-15. As shown in Figure 5, each observation retains its link to the group that it supports.

The groups of function, structure or behavior attributes are used as the basis for determining associations relevant to design synthesis. The observations are used to assign weights to the useful associations. The useful associations for design synthesis are illustrated in Figure 6. Each arrow in the figure is assigned a weight based on the number of observations that support the association. Figure 7 shows an example of computation of a weight for an association between G-F-i and G-S-j.

The groups and their associations provide a basis for structuring design knowledge as design prototypes. The only meaningful associations are those that have a weight greater than one,

indicating that the association occured for more than one observation. The meaningful associations between function and structure groups provide an intial set of partially defined hypothetical prototypes, in which only function and structure attribute-value pairs are specified. The prototypes are further defined using the meaningful associations between function and behavior groups, thereby adding a set of behavior attribute-values pairs to the definition of each prototype. Finally, the hypothetical prototypes are validated by the existence of meaningful associations between behavior and structure groups. The representation of a design prototype as output from DKAO is illustrated in Figure 8.

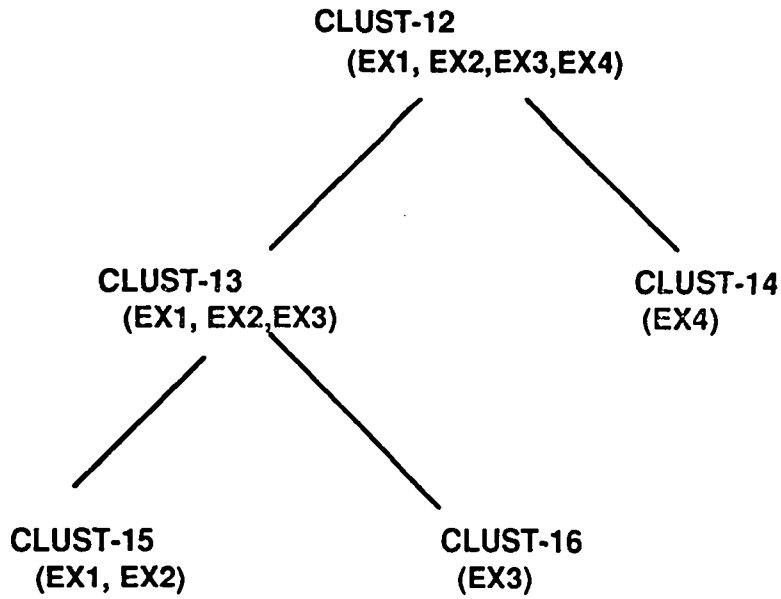


Figure 5: Cluster hierarchy as produced by BRIDGER

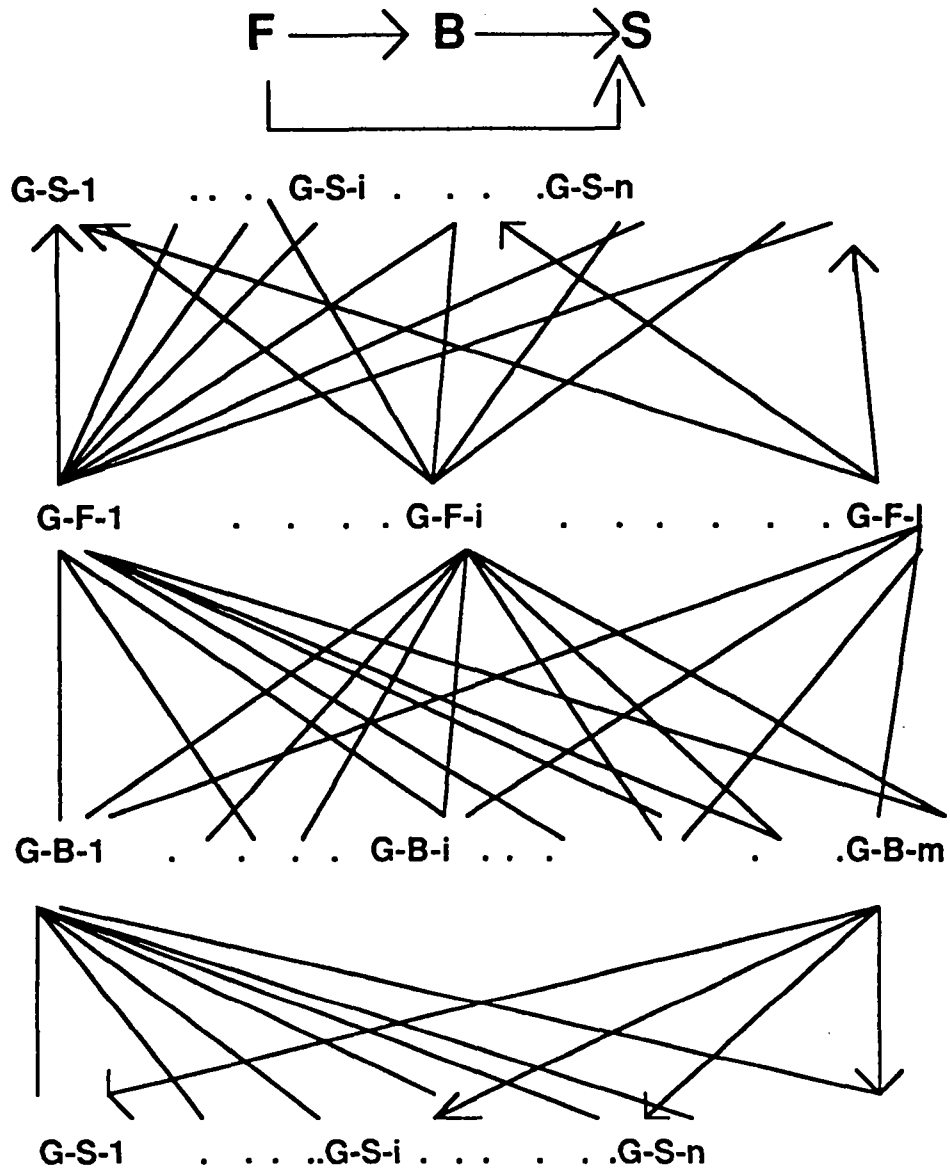
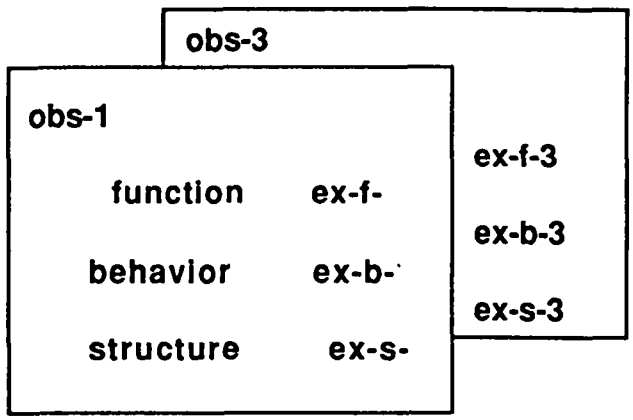


Figure 6: Associations for design synthesis



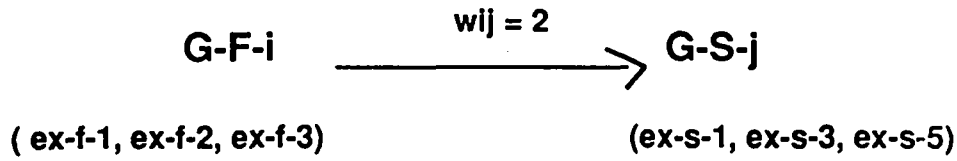


Figure 7: Example of computation of weight for association

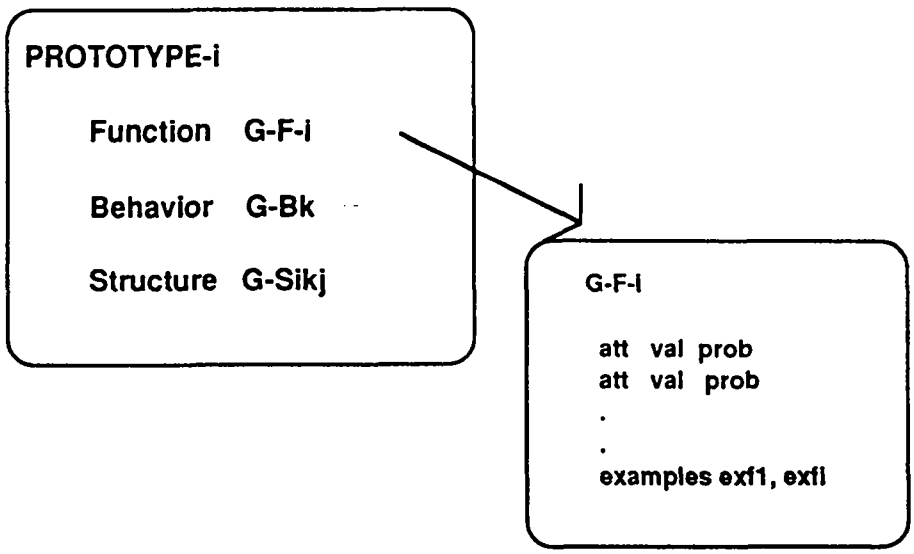


Figure 8: A design prototype as the result of DKAO

5. Application of DKAO

DKAO has been tested using a subset of the bridge data used to illustrate BRIDGER (Reich 1990). The attribute-value pairs of 25 bridges were categorized as function, structure, or behavior attributes. The result of using BRIDGER is three hierarchies; the function hierarchy is shown in Figure 9, the structure hierarchy is shown in Figure 10, and the behavior hierarchy is shown in Figure 11. The number in brackets () next to each node is the number of observations in the cluster.

After the clusters are identified by BRIDGER, the groups are selected as the basis for associations. For the nridge examples there are x function groups, x structure groups, and x behavior groups. Starting from one function group, for example F-G4, we compute the weight for the associations from F-G4 to all structure groups and then from F-G4 to all behavior groups, as illustrated in Figure 12. At this stage, we identify the meaningful associations (those with a weight greater than 1) between the behavior groups and structure groups that are also associated to F-G4.

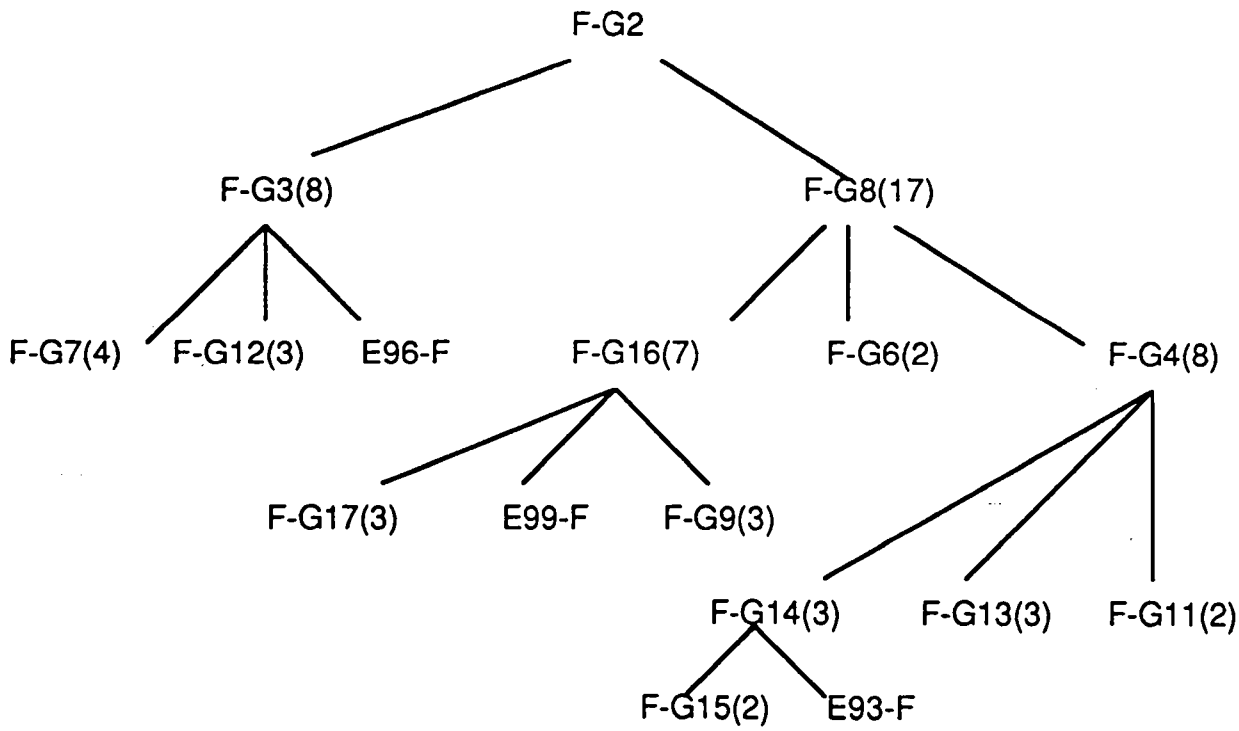


Figure 9: Function clusters for bridge observations

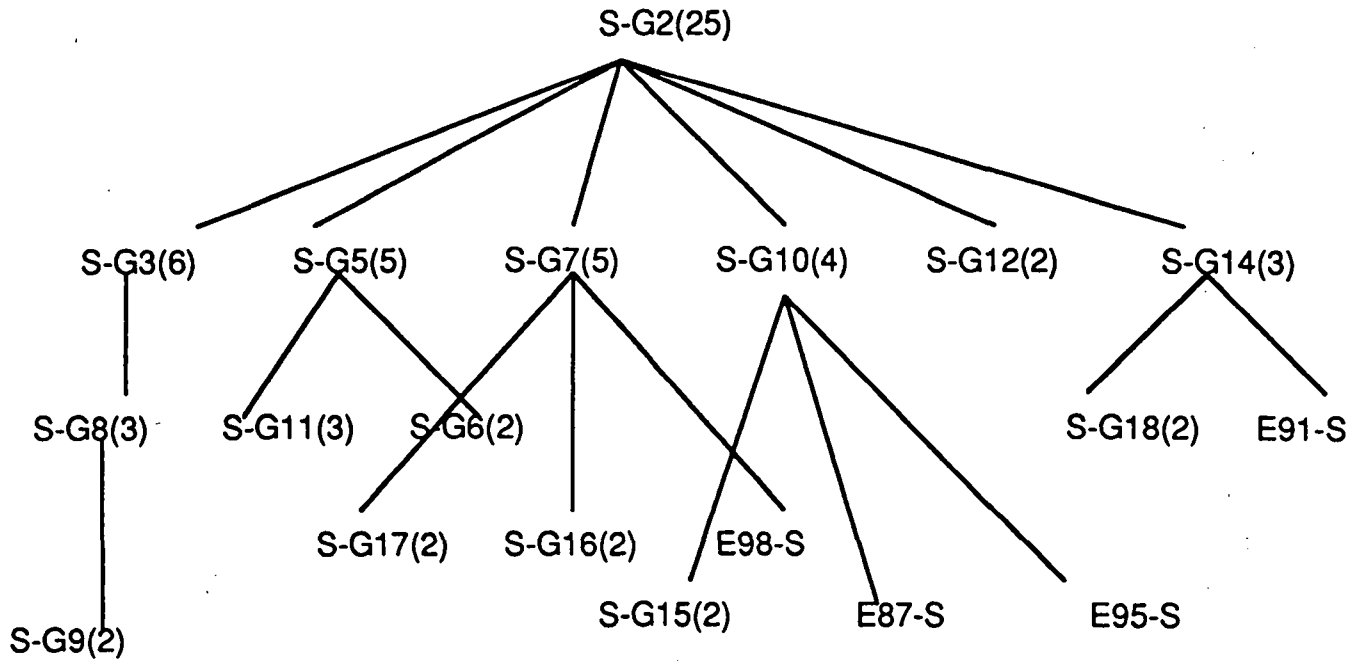


Figure 10: Strucutre clusters for the Bridge observations

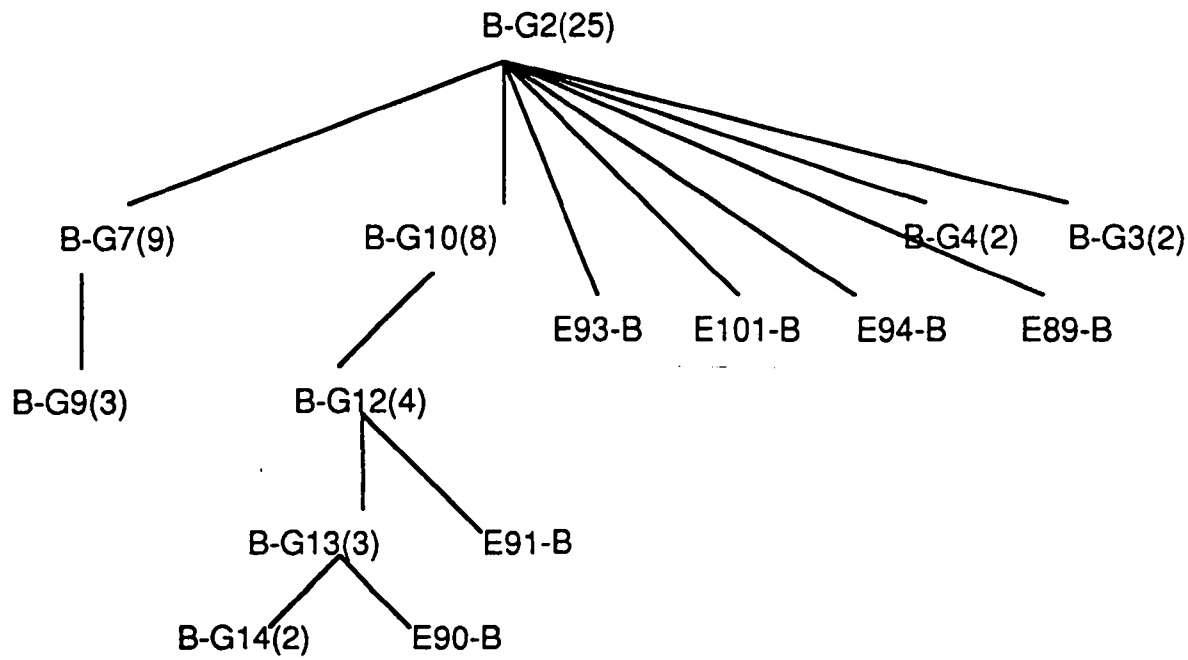
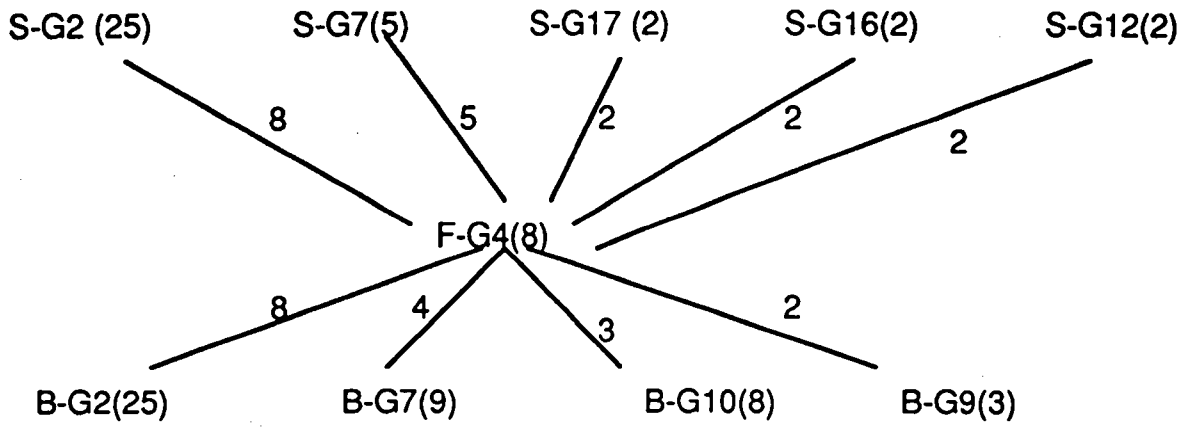


Figure 11: Behavior clusters for the bridge observations



B-G10 associations

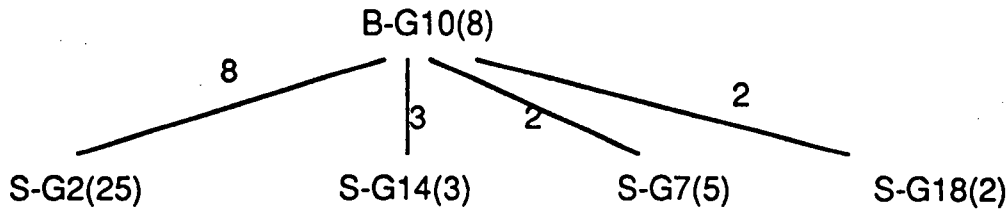


Figure 12: Identifying associations

For example:

F-G4 associations

1. Ass* [F --> S]: (F-G4) = { S-G2 , S-G7, S-17, S-G16, S-G12}
2. Ass* [F --> B]: (F-G4) = { B-G2, B-G7, B-G9, B-G10}

B-G10 associations

3. Ass* [B --> S]: (B-G10) = { S-G2, S-G7, S-G14, S-G18}

The common structure groups in associations numbered 1 and 3 are : S-G2 and S-G7. These common groups validate the following prototypes.

Prototype#100
Function = F-G4
Behavior = B-G10
Strcture = S-G2

Prototype#101
Function = F-G4
Behavior = B-G10
Structure = S-G7

6. Discussion

Although the prototypes identified by DKAO include the relevant categories of design knowledge as defined in (Gero 1990), the prototypes lack any reasoning knowledge for making use of these categories. Upon further inspection of the resulting prototypes, the following can be said.

1. The associations between function, structure, and behavior were generated using observation only, and therefore are no more than heuristics. This occurs because we used domain independent learning algorithms. This is an interesting starting point, but the incorporation of domain knowledge is necessary to make use of these associations.

2. The dependency knowledge among the attributes and their values is not part of the prototype representation. The only knowledge retained from the observations is the most probable value for each attribute.

However, the resulting prototypes are useful as heuristic classifcations of design situations. They comprise a knowledge base that provides a starting point for a design process, in which a selected prototype can serve as an initial design. In many knowledge-based design systems, this is adequate.

7. Conclusions and directions

Our use and extension of inductive learning for design results in a process that produces a set of hypotheses from a set of observations. In our application of inductive learning, we observe design situations and produce design prototypes. The representation of design situations that serve as input to DKAO provide empirical information about the function, structure and behavior of the solutions to design problems. The resulting design prototypes are therefore hueristic generalizations of these situations. This is a useful starting point for the development of a knowledge base for design synthesis. However, such a knowledge base lacks the domain knowledge needed to execute a design process. The knowledge base could be used to predict a starting point for a new design problem, but does not provide the knowledge needed to reason about the acceptability of the design solution. In order to incorporate such knowledge in the prototypes that result from a conceptual clustering approach, additional knowledge about the domain is needed. This requires a shift in the machine learning paradigm applied from an inductive to an analytic paradigm. One technique that shows promise for the continued development of design prototype knowledge is explanation-based learning. This would result in a hybrid approach to learning design knowledge in which empirical, inductive learning is used to formulate a basic generlization of observations and domain dependent explanation-based learning is used to reason about these observations beyond the hueristic associations.

References

Brown, D. and Chandrasekaran, B. (1985). "Expert Systems for a Class of Mechanical Design Activity", in *Knowledge Engineering in Computer-Aided Design*, (editor) J. Gero, pp. 259-283. North-Holland.

Carbonell, J.G. (1990). "Introduction: Paradigms for Machine Learning" in *Machine Learning Paradigms and Methods*, (editor) J. Carbonell, MIT/Elsevier, pp.1-10.

Chandrasekaran, (1990), "Design Problem Solving: A Task Analysis" in *AI Magazine*, Winter.

Reich, Y. (1990). "Design Knowledge Acquisition: Task Analysis and a Partial Implementation", in *The 5th Knowledge Acquisition for Knowledge-Besed Systems Workshop*, Banff, Canada.

Feigenbaum, E.A. and Simon, H. (1984). "EPAM-like Models of Recognition and Learning", *Cognitive Science*, 8, pp. 305-336.

Gluck, M. and Corter, J. (1985). "Information, Uncertainty and the Utility of Categories", in *Proceedings Seventh Annual Conference of the Cognitive Sciences Society*, Ivrine, CA. pp. 283-287.

Fisher, D.H (1987). "Knowledge Acquisition via Incremental Conceptual Clustering " in *Machine Learning*, 2, pp. 139-172.

Forgy, C.L. (1981). "OPS5 User's Manual", Technical Report CMU-CS-81-135, Carnegie Mellon University, Pittsburgh PA.

Gero, J.S., Maher, M.L. and Zhang, W. (1988). "Chunking Structural Design Knowledge as Prototypes" in *Artificial Intelligence in Engineering: Design*, (editor) J. Gero, Elsevier/Computational Mechanics Publications, pp 3-21.

Gero, J. (1990), "Prototypes: A Knowledge Representation Schema for Design" in *AI Magazine*, Winter.

Lebowitz M. (1987). "Experiments with Incremental Concept Formation : UNIMEM " in *Machine Learning*, 2 , pp. 103-138.

Maher, M.L. and Fenves. S.J., (1984). "HI-RISE: A Knowledge-Based Expert System for the Preliminary Structural Design of High Rise Buildings", Technical Report , R-85-146, Department of Civil Engineering, Carnegie Mellon University.

Maher, M.L. (1988). "Engineering Design Synthesis: A Domain Independent Representation" in *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 1(3).

Maher, M.L. (1990). "Process Models of Design Synthesis" in *AI Magazine*, Winter.

Marcus, S., Stout, J., and McDermott, J. (1988). "VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking". in *AI Magazine* 9(1), pp. 95-114.

McDermott, J. (1980). "R1: A Rule-Based Configurer of Computer Systems", Technical Report CMU-CS-80-119, Carnegie Mellon University, Pittsburgh PA.

Michalski, R.S. and Kodratoff, Y. (1990). "Research in Machine Learning; Recent Progress, Classification of Methods, and Future Directions", in *Machine Learning An Artificial Intelligence Appproach Volume III*, (editors) Y. Kodratoff and R. Michalski, Morgan Kauffmann, pp.3-30.

Mitchell, T.M., Steinberg, L.I., and Shulman, J.S. (1984). "A Knowledge-based Appproach to Design" in *Proceedings of the IEEE Workshop of Principles of Knowledge-based Systems*, IEEE pp 27-34.

Nyberg, E.H. (1988). "The Framekit User's Guide Version 2.0", Technical Report CMU-CMT-MEMO, Carnegie Mellon University, Pittsburgh PA.

Tham, K.W., Lee, H.S., and Gero, J.S. (1990). "Building Envelope Design Using Design Prototypes" in *AI in Building Design: Progress and Promise*, ASHRAE Symposium, St. Louis, Missouri.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique

ISSN 0249-6399