1988

Using analogical reasoning to design buildings

Fang Zhao
Carnegie Mellon University

Mary Lou. Maher

Carnegie Mellon University.Engineering Design Research Center.

Using Analogical Reasoning to Design Buildings

by

F. Zhao and M.L Maher

EDRC-12-22-88

USING ANALOGICAL REASONING TO DESIGN BUILDINGS

F. Zhao and M.L Maher
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, PA

Abstract

This paper presents a methodology for identifying the relevant design elements for the synthesis of new structural designs using previous design situations and their corresponding solutions. The study is a reflection of the observation that engineers use related experience when solving new problems. The methodology is an application of transformational analogy, a form of analogical reasoning.

A prototype system STRUPLE has been developed to implement the methodology, making use of knowledge based expert system techniques. The emphasis of STRUPLE differs from that of traditional expert systems in that the latter only use formalized or compiled knowledge while STRUPLE uses an experience database as a knowledge supplement.

1. Introduction

Engineering design is a creative process in which the experience and knowledge of the designer serve as resources. Facing a new design task, an experienced designer will easily recall similar cases which he or she has met before, and will attempt to make an appropriate adaptation of one or more previous design solutions to fit the new situation. A less experienced designer may go through much trial and error before he or she achieves a suitable design solution. The difference between the two is experience, which can only be accumulated over a long period of practice. This paper proposes a methodology for using previous design situations and their solutions to plan the synthesis of new designs. The area of application is the preliminary structural design of buildings.

A popular way of using experience to solve new problems is to develop an expert system that contains an appropriate representation of the relevant experience. MYCIN [Buchanan 84], DIPMETER [Davis 81], VT [Marcus 86], XCON [Kraft 84] and VEXED [Mitchell 85] are examples of such expert systems. Most current expert systems use experience in the form of heuristics that are well recognized and have become rules of thumb. Such heuristics arise from the experience gained from occurrence of related situations and are usually summarized by experts in their particular domains. This means that such expert systems use a very small portion of the experts' experience, since most experience is not easily represented by heuristic rules. In order that more of the experts' experience can be utilized, and that through the use of such less formalized experience one can learn their implications so that one can discover heuristics or even the principles behind them, there is a need to find a way to use experience in computer programs directly, instead of limiting them to using compiled knowledge only.

A current structural design expert system, HI-RISE [Maher 85], relies on the definition of a knowledge base containing objects representing structural subsystems and rules representing the appropriate way to combine these objects and compare the resulting solutions. The methodology described in this paper expands the knowledge used during the structural design process to include experience in the form of previous building design situations and solutions. Specifically, this knowledge is used to identify the relevant search space for the synthesis of alternative structural systems for a given building design problem.

This paper is organized into five sections. The second section introduces analogical reasoning as a problem solving approach. The third section briefly discusses structural design and highlights the phases relevant to this paper. The fourth section describes a prototype implementation for using design experience directly. Finally, the last section provides some conclusions.

2. Analogical Reasoning

According to Paul Harmon and David King, knowledge can be considered in two categories: *deep knowledge* and *surface knowledge* ([Harmon 85]). *Deep knowledge* includes general theories that are derived from domain-independent facts, definitions, first principles, axioms and laws. *Surface knowledge* includes domain and performance theories that evolve from domain-dependent facts and their generalized forms - heuristics. Since engineering design theories are not well developed and engineering design is mostly empirical, especially during the early design stages or in unprecedented designs, engineering design processes involve more surface than deep knowledge. Moreover, domain theories are sometimes insufficient to guide design processes in producing successful or optimal solutions, and heuristics and experience have to be used. This explains why in the engineering design field, though not the only field, experts and experience are so cherished.

Another way to view knowledge is in terms of the degree of its compilation. Compiled knowledge is "information that is organized, indexed and stored in such a way that it is easily accessed." [Harmon 85] The knowledge used to analyze and verify an engineering design is compiled to the degree that it can be expressed in algorithms. An example is the finite element method. The design heuristics are often also compiled in the sense that they are well known and easily indexed. Experience that is used by people consciously or unconsciously is not compiled; this experience can sometimes give inspiration and lead to creative thinking. The advantage in using experience directly is that they contain richer information than heuristics and theories and can be helpful when no heuristics and theories are available. However, uncoupled experience usually contains also irrelevant information that can be misleading. Therefore, irrelevant information contained in the experience has to be filtered out and the conclusions drawn from them need to be carefully verified.

Analogical reasoning is a very efficient way to use past experience. In an analogical reasoning process, previous experience of similar situations is recalled, related and selected to solve a new problem. People use analogical reasoning every day solving various problems and are very successful. It is suggested by Carbonell that analogical reasoning is "a central inference method in human cognition". The following definition of analogical reasoning is given by Carbonell [Carbonell 86]

> **Definition:** *Analogical problem solving consists of transferring knowledge from past problem solving episodes to **new** problems that share significant aspects with corresponding past experience and using the transferred knowledge to construct solutions to the new problems.*

The two types of analogy of interest here are:

1. **Transformational analogy.** For old and new problems, if the problem statements have strong similarity and their solutions fall in the same category, the past solutions can be transferred, i.e. retrieved, modified and augmented, to satisfy the criteria of the new problems. [Carbonell 83a, Carbonell 86]
2. **Derivational analogy.** Sometimes, even though old and new problems have similar problem statements and problem solving process or techniques, the resultant solutions may bear little direct similarity. In such cases, the reasoning steps in the construction of the past solutions may be retrieved and modified in order to construct derivational paths toward the solutions to the new problems [Carbonell 86].

Transformational analogy is only concerned with the problem definitions and their solutions and ignores the reasoning processes that result in those solutions. Machine learning in this case requires storing the solution to the new problem in memory to be used later. However, transformational analogy cannot provide the reasons for the decisions made, which in many cases are relevant and important to solving new problems.

Depending on the nature of the problem at hand, derivational analogy is sometimes more suitable. Some problems that appear very different may have similar goal structures, or use similar reasoning steps. In this case, the reasoning steps can be replayed, evaluated whether they are applicable to the new problem, and if so, modified as necessary. Argo[Huhns 87], developed recently by the

Microelectronics and Computer Technology Corporation, is a system that employs this approach for solving generic design problems. This approach requires more information to be stored, namely not only the problem statements and final solutions, but also all the intermediate states of the problem solving, including how decisions are made under the current conditions, how the alternatives are evaluated, accepted, or rejected, and how a path is followed and for what reason it fails. Derivational analogy stresses the problems' similarity on logic rather than on particular problem characteristics, thus it is very powerful for applying the same knowledge to problems in different domains. However, this approach usually does not allow discontinuity in reasoning, which is a characteristic of innovative design. Despite the power and usefulness of this approach, it is more difficult to implement for structural design than the transformational analogy, because both the identification of problems which may have the same reasoning processes and the formulation of the reasoning processes are very difficult using the current AI theories and techniques. For now, our interest is in the transformational analogy, because it is more easily applied to structural design and can serve as a good start in this study.

Regardless of which analogical reasoning approach is used, the following important issues have to be addressed and settled before any implementation attempt [Carbonell 86].

1. What are the "significant aspects" shared by old and new problems? In other words, the similarity between old and new problems must be identified. The nature of this similarity will determine whether the analogical reasoning approach is suitable and, if so, whether transformational analogy or derivational analogy is to be used. The identified common aspects that the old and new problems share also serve as the basis of a similarity metric. This similarity metric is to be used to search for the solutions to or reasoning steps of old problems which are relevant to the new problem solving.

2. How is the past successful experience to be selected from a possibly large long term memory? Retrieving past experience is a search process in which the solutions to old problems are examined and measured by the similarity metric, and then selected or rejected. Because the search space for the analogical reasoning approach is potentially very large, the efficiency of search is of concern. The efficiency is determined mainly by the way in which past experience is represented and stored. To achieve search efficiency, experience must be organized and stored in such a way that during the search only the most relevant experience is pursued, irrelevant experience is ignored, and other experience is considered only when it becomes necessary.

3. What knowledge is to be transferred from the past experience to the new solutions? The answer depends on the nature of the problem to be solved, the type of the analogical approach used, and the results of past experience retrieval. The nature of the problem determines the knowledge about a particular domain that is needed for constructing its solution. The use of transformational analogy or derivational analogy specifies the source of the knowledge that will come from either the solutions to the old problems or the reasoning steps used in past problem solving. Finally, the result of past experience retrieval will show the availability and suitability of the knowledge represented by the experience. The knowledge can be in a very abstract and comprehensive form as solutions to a problem or a subproblem solved before, or it can be in a concrete and simple form as a physical component used to construct a mechanism, a rule of thumb used to make a decision, an algorithm applied to produce a result under certain conditions, etc.

4. How does the knowledge transformation process occur? Transferring the knowledge from past experience to the solution of a new problem or to the process of constructing a solution of a new problem is a problem solving process itself. In simple cases, algorithms, functions, or perhaps heuristic rules can be used to convert the knowledge found in the past experience into a constructive component of the new problem solving task. In more complicated cases, use of various problem solving strategies becomes necessary.

Of the above four issues addressed, the first and last ones are usually the most important and the most difficult. We will illustrate these ideas in more detail by describing STRUPLE, an implementation of transformational analogy, in Section 4.

3. Building Design

Due to the complexity of modem buildings, it is common practice that the architectural and structural design are performed by different professionals. Usually, architects conceive the form of the building, assign functions to different spaces in the building according to the intended floor uses, give the spaces geometric attributes, etc. Structural design is then performed by engineers to give a precise description of the physical form of the building that meets the architect's requirements. In this section, we briefly discuss architectural specifications and their impact on structural design, and possible structural design methods suitable to the use of analogical reasoning.

3.1. Architectural Specifications

The architectural design products are the architectural specifications, usually in the form of drawings, which specify mainly the geometric, topological, and functional attributes of a building. Sometimes, the specifications also contain the architect's preferences for some particular construction material or structural systems for architectural purposes. In current practice, structural engineers usually take architectural specifications as a problem statement that partially specifies the building structure, depending on how explicit the specifications are and whether the building design is common or unusual. These specifications implicitly define a subspace of the design space in which the structural engineer searches for solutions in the form of structural systems that satisfy the architectural specifications.

3.2. Structural Design

Given the architectural specifications, the structural design process starts with the definition of a need to transmit loads in space to a support or foundation, subject to constraints on cost, geometry, and other criteria. The final product of the design process is the detailed specification of structural components capable of transmitting these loads with the appropriate levels of safety and serviceability. The design process may be viewed as a sequence of three stages: preliminary design, analysis and detailed design.

1. PRELIMINARY DESIGN involves the synthesis of potential structural systems satisfying a few key constraints, and the selection of one, or at most a few, systems to be pursued further. Synthesis requires a knowledge of structural subsystems and their appropriateness for different situations.

2. ANALYSIS is the process of modeling the selected structural system and determining its response to external effects. This process involves transforming a physical structure to a mathematical model, analyzing the model, and interpreting the results of the analysis in terms of the actual physical structure.

3. DETAILED DESIGN is the selection and proportioning of the structural components such that all applicable constraints are satisfied.

The first stage, preliminary design, deals with a different level of abstraction than analysis and detailed design. During preliminary design, the form of the design solution is identified. During analysis and detailed design this form is refined. There may be significant deviations between the properties of components assumed at the analysis stage and those determined at the detailed design stage, which necessitates a reanatysis. Other major and minor cycles of redesign may also occur. The process continues until a satisfactory (or optimal) design is obtained. The *conceptualize-analyze-detall* cycle is typical of many design paradigms.

Many of the conceptual aspects of structural design are embodied in the preliminary design phase. At this point, the only information available to the designer are the architectural specifications and constraints on the solutions. It is during this stage of the design process that the creativity and experience of the engineer are mostly needed. The increasing complexity of engineering design problems has made synthesis a very difficult process, even for an experienced designer, if not approached in a structured and organized fashion.

There is no standard approach to the synthesis process suitable for all design problems. One approach is to decompose the design problem into the design of independent or loosely-coupled

subsystems. In a similar manner, each subsystem can be further decomposed into major components. The subsystems and components associated with structural engineering form the design space in which the synthesis process operates.

3.3. Design Space, Design Vocabulary and Experience

The synthesis of alternative designs can be considered as the manipulation of a design vocabulary that forms a subset of the design space. A design vocabulary is a set of design elements that may be used as the constructing components of a certain class of products. For structural engineering, a design element can be a basic structural component such as a beam or column, or it can be a structural subsystem such as a rigid frame or braced frame, which are obtained by combining the basic components according to certain well accepted rules. The design space for a particular class of problems includes the description of the levels of abstraction and of the subsystems and components the designer knows about. For example, the design space for the structural system for buildings includes knowledge about rigid frames, braced frames, shear walls, etc. The boundary of the subset of the design space appropriate for a given situation defines the design vocabulary to be considered and identifies the design knowledge required.

The boundary of the subset of the design space is often unclear given only the architectural specifications, and it varies with different structural engineers depending on their knowledge and experience. Identifying the boundary of a design subspace is not an easy task but is very important. It affects the efficiency and quality of the subsequent synthesis. An experienced and knowledgeable engineer not only possesses knowledge of a large design space, but also is very good at determining the most appropriate design vocabulary for a given situation as well as the knowledge needed for the synthesis. This kind of experience is difficult to capture in an expert system that contains heuristics in rule form.

4. STRUPLE

STRUPLE (STRUctural PLanning from Experience) is a prototype system intended to demonstrate how the analogical reasoning approach can be applied to structural design. The effort has focused on the representation of experience, the development of a similarity metric to find relevant experience, and the definition of criteria for selecting the design vocabulary. In this section, the concepts used in STRUPLE and their applications are presented. Except for some basic ideas about analogical transformation and its major aspects, the rules and formulas used in STRUPLE come from the authors' own knowledge about structural engineering, rather than from any publications or structural design experts.

4.1. Overview of STRUPLE

STRUPLE is not a complete system that performs the structural synthesis. Instead, it serves as a pre-processer for a structural design synthesizer. STRUPLE accepts the description of a building to be designed, searches relevant past experience and determines the design vocabulary to be used in the synthesis according to the useful experience found.

The specification of the current building to be designed includes the following information:

- the location of the building and the project budget;
- geometric information such as the gross area, dimensions, shape, etc.;
- architectural information about the number of stories, intended floor use, typical bay sizes and their numbers in different directions, and the number and location of the service shafts; and
- loading information including the design live load, the design wind load, the seismic zone, and the soil bearing capacity.

Some information is not used by STRUPLE, e.g. number of bays and number of service shafts, but is included because it is needed during synthesis. In addition to the specification listed above, STRUPLE

also checks if the user wants to specify an initial boundary on the design space by choosing among alternatives of construction materials, 3D systems, 2D systems, floor systems and foundation systems.

The output from STRUPLE is the design vocabulary selected for synthesis. STRUPLE can be linked to a synthesis program which will take the results of STRUPLE as input and synthesize the structural configuration using the design vocabulary provided.

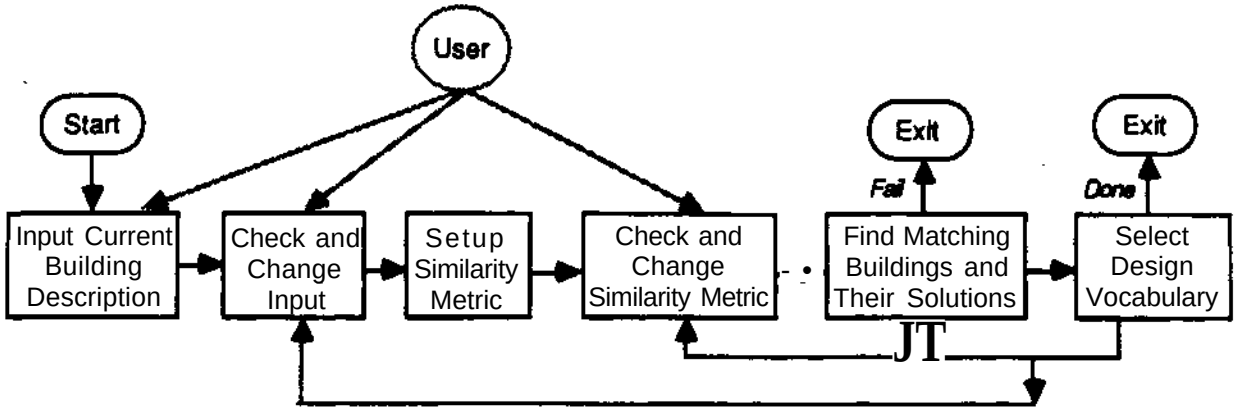

Figure 4-1: Overall Process of STRUPLE

The overall process of STRUPLE is shown in Figure 4-1. STRUPLE starts a working session with the input described above. The similarity metric is then set up using the input. The user can check and change it. After the user approves the similarity metric, STRUPLE uses the similarity metric to find matching buildings stored in a database as experience. If no matching buildings exist, the user can either modify the similarity metric used in the previous search, or modify the input information about the building and the similarity metric will be redefined according to the modified input. Any change of the similarity metric will initiate a new search process. If there are matching buildings, STRUPLE examines their design solutions and selects the design vocabulary from these solutions for the new design.

4.2. Representation of Experience

Experience is stored declaratively in STRUPLE in a database, also called the experience base, and is represented in the form of incomplete descriptions of building design solutions, which give the following information about buildings.

1. General information: Building number, name, design completion date, city and state where it is located, owner, general use of the building, and unit and total costs (land cost is excluded).

2. Geometric information: Gross area, height, width, aspect ratio, overall shape and shape irregularity.

3. Other architectural specifications: The number of stories below and above grade, their intended floor uses, bay numbers and sizes in the two principal orthogonal directions, floor-to-floor height, floor-to-ceiling height, number of service shafts and general description of their location in the build plan.

4. Loading information: Dead load, live load, wind load and seismic zone.

5. Primary 3D systems: A building may have more than one type of 3D system, for example, one or more cores with a 2D orthogonal system, which may be composed of rigid frames, tied together.

6. Lateral 2D systems: Lateral 2D systems are the subsystems of 3D systems. The information for each lateral 2D system includes the structural type, the construction

material, and direction of the placement.

7. Floor system type: In addition to structural information, system type implies material type, support system, and support system material type.
8. Foundation system type: System type provides structural information, material type, and bearing capacity.

In order to determine whether the design solutions represented in the experience base were successful, they should be evaluated by experts not only from the viewpoint of structural engineers, but also those of other professionals participating in the project. The evaluation should be made on a holistic basis including aesthetics, function and economy of the design solution. Here, we assume that all the buildings which are stored in the database are successful designs in order to simplify the problem.

4.3. Selection of Experience

In STRUPLE, the selection of experience involves finding the relevant experience from the database and ranking them through evaluation. The result is an ordered set of design elements that identifies the subset of the design space for a given structural design problem.

4.3.1. Finding Relevant Experience

In STRUPLE, relevant experience for a new building is a set of similar buildings found in the database. The buildings which are considered by STRUPLE to be similar to the current building are called *matching buildings,* or sometimes *matches,* and are identified using a similarity metric. In STRUPLE, the similarity metric is described by a set of *criteria,* also called *matching criteria,* which define what significant common aspects the matching buildings and the current building should share. A matching criterion is a requirement of similarity imposed on a *feature* of a matching building. For example, the number of stories, the intended use, the design wind load, the construction material, etc., all are features of a building and are potential criteria. Obviously, not all features of a building are equally important in terms of their impacts on the choice of a structural system for the current building. Thus, criteria are divided into three classes: *required criteria, desired criteria* and *no-match criteria.* Required criteria must be satisfied by a matching building. The satisfaction of desired criteria is desirable but not necessary. No-match criteria are not considered. *Required, desired* and *no-match* are qualifiers that are defined as the *status* of criteria.

The status of a criterion is assigned by STRUPLE according to the description of the current building. Whether a criterion is defined as required or desired depends on the extent to which it would aid in determining the structural system for the current building. For instance, when a building has over thirty stories, the lateral system design will more likely govern the design, or at least be as important as the gravity system. In this case, wind load is chosen to be a required criterion. If the the number of stories of the current building is in the range of five to thirty, wind load will not be as important a factor and is defined as a desired criterion. When the building has less than five stories, wind load will not govern the design, so it is defined as a no-match criterion. Table 4-1 lists all the criteria used in STRUPLE as well as their status assumed in different cases.

When searching the database for similar buildings, required criteria are used as qualifiers. In order to locate a matching building explicitly, the limits within which the building can be considered to be a match must be known. Thus a *range* must be set for each required criterion. A range is defined by the limits of allowable difference of the feature values of the two buildings compared. A required criterion is satisfied by a building being matched if the value of its corresponding feature falls within the range of the criterion. There are features which do not have numerical values but can be discretized; examples are intended use and construction material. For matching these features, *check-lists* are used. A required criterion on such a feature has one or more items on its check-list, each of which is an alternative that is considered to be an eligible value of a feature. When a required criterion has a check-list, it is satisfied by a building if the feature value of the building is an item on the corresponding check-list.

Criteria ranges are set heuristically according to the feature values of the current building. For most

Criteria	*Status*	
unit cost	required	if project budget is tight
	desired	if project budget is not tight or unknown
state	desired	
stories above grade	required	
stories below grade	required	
intended use of stories below grade	required	if number of stories below grade > 0
	no-match	if number of stories below grade = 0
intended use of stories above grade	required	
building shape	required	
typical bay size	required	if intended use is commercial
	desired	if intended use is residential
floor-to-floor height	required	
floor-to-ceiling height	required	
wind load	required	if stories >* 30
	desired	if 5 < stories < 30
	no-match	if stories <» 5
live load	required	
seismic zone	required	
soil bearing capacity	required	if that of current building available
	no-match	if that of current building not available
construction materials	required	if user has preferences
	no-match	if user does not have preferences
3D systems	required	if user has preferences
2D systems	required	if user has preferences
floor systems	required	if user has preferences
foundation systems	required	if user has preferences

Table 4-1: Criteria and Their Status

building features with numerical values, e.g. floor-to-floor-height, live load, etc., ranges are set arbitrarily to be from 0.85 to 1.15 times the corresponding values for the current building. The exceptions are unit cost, seismic zone and number of stories, in which the authors have enough information to be more discriminating. Criteria that have check-lists instead of numerical ranges are intended use, building shape, and those on the structural subsystems and construction materials. For criteria on structural subsystems and material types, the check-lists contain the subsystems or material types that the user prefers to use. For others, the corresponding feature values for the current building constitutes the check-lists. Two particular extensions are made to the intended use when the current building has more than 40 stories and the intended use is residential or commercial. In such cases, both commercial and residential are considered similar and appear in the check-list for intended use. The criterion on building shape usually requires that a matching building have the same shape as the current building. However, when the current building's shape is multi-rectangle, e.g. L shaped or U shaped, rectangle is considered to be an additional matching alternative.

Before STRUPLE searches the database for matching buildings, the user is allowed to check the criteria. This includes changing the status of a criterion, redefining the matching range for a criterion, and adding or deleting matching alternatives on the check-list for a criterion. However, certain criteria's status can not be changed by the user in order to ensure that the matching criteria make sense. For example, the status of criteria on intended use and number of stories above grade must always be required. A matching alternative can not be removed if it is a feature value for the current building.

After the user approves all the criteria, STRUPLE searches the database to find all the matches which satisfy the required criteria. If matches are found, STRUPLE presents them to the user and allows the user to determine if the matching buildings found by STRUPLE are in fact similar to the current building. Then, STRUPLE proceeds with the similarity evaluation of the matching buildings if the user approves any of the matches.

4.3.2. Ranking Relevant Experience

Each matching building is evaluated and ranked to measure how well it resembles the current building according to both required and desired criteria. The method for evaluation is intuitive and simple; it is similar to measuring the relative error of two function values. This method requires that desired criteria also have ranges as required criteria. The ranges for desired criteria are defined in the same way as those for required criteria. When used for evaluation, the ranges indicate desired limits within which the values of different features of the matching buildings are expected to fall, though a value's falling out of a range will not cause the elimination of a matching building.

The evaluation of an individual matching building is completed by *feature evaluations* and a *global evaluation.* A *feature evaluation* is made on each of the features that have required or desired criteria, and is a measure of the difference between the feature values of the current building and a matching building. The result of a feature evaluation is a real number between 0.0 and 1.0. A positive value is considered a penalty in the sense that 0.0 means an exact match while 1.0 means no match or that the difference is too large. A *global evaluation* is a comprehensive measure combining all evaluations on individual features of a matching building.

Feature evaluations are done in different ways, depending on whether a criterion has a numerical range or a check-list. The following formula is used for a feature evaluation of a criterion with a numerical range:

$$e \qquad \wedge \qquad \wedge \qquad 1 \qquad \frac{\text{rea}}{} \tag{1}$$

where e_{lk} » evaluation of feature *i* for match *k*
v_{ik} = the value of feature *i* for match *k*
$v_{cuntntjL}$ [a the value of the same] feature for the current building
u_i = the upper limit of the range for criterion i
/•» the lower limit of the range for criterion i

If the result is greater than 1.0, it is set to 1.0.

A feature evaluation of a criterion with a check-list is made according to the following rules.

IP v_{ik} is in the check-list
AMD v_{ik} is the same as $v_{currgnifi}$
THEM e_{ik} is 0.0

IP v_{ik} is in the check-list
AND v_{ik} is not the same as $v_{currenii}$
THEM e_{ik} is 0.4

IP v;* is not in the check-list
THEM e_{ik} is 1.0

Here $e_{\pm\%}$ v^ and $*_{cwrma\%;}$ have the same meanings as defined before, but they are strings of characters instead of numerical values.

Cost is an important factor in the selection of structural systems. If the user desires, cost is included as another feature evaluation. In the database, the unit cost of each building is stored (i.e., as $ psf). The evaluation is made differently in two cases. In the first case, the estimated project budget is given and STRUPLE calculates the estimated unit cost by dividing the estimated project budget by the gross area of the current building and determines a matching range for the unit cost. Formula (1) is used in this case to evaluate each matching building based on the unit cost. In the second case, the project budget is not given and the cost evaluation is made on a relative basis. The maximum unit cost and the minimum unit cost of all matching buildings are found and STRUPLE uses these values to evaluate each match with the following formula:

$$e_{costJt} = \frac{cost_k - cost}{cost\_ - c\ cost^{\wedge}} \quad (2)$$

where e_{costJt} «the cost evaluation for match k
$cost_k$ »the unit cost of match k being evaluated
^{cost}min = the minimum unit cost among all the matches
$cost_{max}$ = the maximum unit cost among all the matches

The global evaluation of a matching building is the summation of all feature evaluations multiplied by the corresponding weights in a weight table defined in STRUPLE. The weights reflect the degree of relative importance of the criteria. Each criterion has two weights; one to be used if the criterion* status is required and one if the status is desired. This is due to the fact that for a required criterion, the weight should be greater than that corresponding to a desired criterion.

$$e_k = \sum_{i=1}^{m} e_{i*} \quad (0)$$

where e_k = the global evaluation for match k
e_{ik} = feature evaluation i for match k
w_i - the weight for feature i
m = the total number of the features evaluated

Ranking of the matching building is done according to the global evaluation values of the matching buildings. A match which has a smaller evaluation is ranked as a better match since a smaller evaluation value indicates less difference **between** the match and the current building. The rank of a match is indicated by an integer. The purpose is to provide a clearer view to the user of how one match compares with the others.

4.4. Structural Configuration Planning Using Selected Experience

The third subproblem of analogical transformation is to determine what knowledge is to be transformed. Even though two buildings could **have** many common aspects, because of their different special design requirements, the design solutions may not be the same. This means that it is not always feasible to take the whole solution for one building and convert it into one for the other. However, a certain group of design vocabulary frequently appears in certain types of buildings. Although we can not directly use the design solution of an existing building for a new building, we can use the design vocabulary of the old design in constructing the new solution. Thus the design vocabulary is chosen as the knowledge to be extracted from the old design solutions and be transformed.

The last task in the analogical approach is to embody the transformation of knowledge existing in the past experience. Here, the transformed experience is considered to be the different types of structural elements used in similar buildings. STRUPLE has a set of structural elements that are stored in its knowledge base representing the complete design space that the synthesis process knows about. In STRUPLE, a building is decomposed into several levels of abstraction (Figure 4-2). The elements of design space are organized into different levels of abstraction under lateral system and gravity system

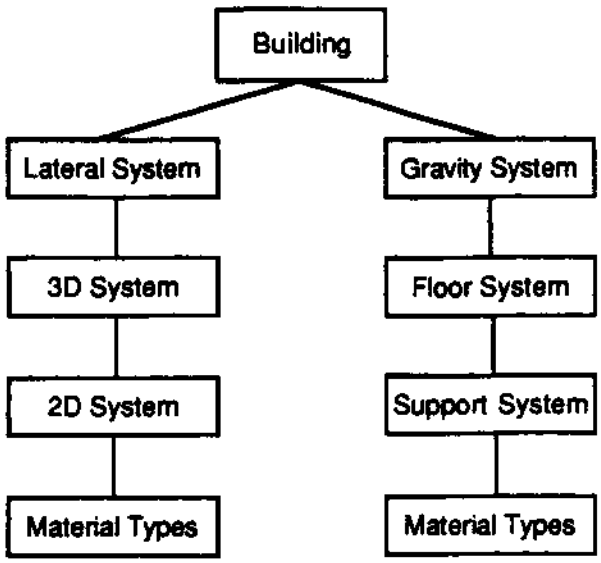

Figure 4-2: Levels of Abstraction of a Building

Levels	*Alternatives*
3D system	tube tube in tube core core and suspension bundled tubes 2D othogonal
2D system	rigid frame braced frame megaframe solid wall staggered trusses
material type	masonry steel high strength steel precast re prefab-prestressed concrete cast-in-place re cast-in-place prestressed re cast-in-place post-tensioned re

Table 4-2: Design Space for Lateral Systems

levels, as shown in Table 4-2 and Table 4-3.

For each level of abstraction, STRUPLE determines the design vocabulary to be considered for the new building, thus identifying a subset of structural elements that is most promising. This design vocabulary will be used during the synthesis process, in which each structural element will be examined to determine whether it is an eligible or efficient alternative.

The elements of the design vocabulary selected by STRUPLE should come from the matching buildings. In order to make the use of design vocabulary efficient, the elements in the design vocabulary are ordered by STRUPLE. First, the occurrence of each element of design vocabulary used in the old

Levels	*Alternatives*
floor system	flat slab beam and slab waffle grid plank steel deck and concrete topping open-web joists and composite concrete
support system	column beam and column beam-girder and column wall
material type	masonry steel high strength steel precast re prefab-prestressed concrete cast-in-place re cast-in-place prestressed re cast-in-place post-tensioned re

Table 4-3: Design Space for Gravity Systems

design solutions is calculated. Then each element is assigned a priority of consideration according to its occurrence in the matching buildings and the global evaluations of those matching buildings in which it is used. A high priority indicates that this type of element is frequently used in the buildings similar to the current one and should be considered first.

There are several general situations when selecting design elements. First, if all types of elements at one level have close priorities, all of them should be considered one by one during the synthesis. Second, if a few types of elements have much higher priorities than the others, then those with very low priorities may not be considered if satisfactory solutions can be obtained by using the elements with high priorities. Third, if no similar building is found, or there are only a few types of elements at each level with very low priorities, then all structural elements in the design space should be considered. In the third case, useful experience is not available and STRUPLE has to use the general design knowledge stored in its knowledge base.

The computation of priorities and the selection of design elements is done in the following steps:

1. Compute the occurrence o_{ijk} of an element j at level of abstraction i that is present in a single matching building k and this element's priority p_{ijk} concerning this single building. The priority is calculated as follows

$$p_{ijk} = \frac{o_{ijk}}{1 + e_k} \tag{4}$$

where e_k is the global evaluation for matching building *, computed from Formula (3). o_{ijk} should be 0 or 1, where 0 indicates that element j does not appear in building k and 1 the opposite. It can be seen that the priority of an element increases when the value of the global evaluation of the corresponding building decreases, which indicates a better match for the input building.

2. Compute o_{tJ}, the total occurrence of an element j at level of abstraction i in all matching buildings and p_{tJt} the sum of the priorities of this element computed from Equation (4). o_{tj} indicates the number of matching buildings in which element/ has been used. It is an integer between 0 and $n_{\%}$ where n is the total number of matching buildings. If o_{ij} is 0, then element j is not used in any of the matching buildings. If o_{tj} is *, it is useo in all the

matching building. o_{ij} will be used when determining whether element; should be included in the subset of design vocabulary for the synthesis. p_{ij} has the same property as p_{ijk}: it increases when o_{ij} increases or the evaluations of the buildings in which it appears decrease. After normalization, p_{ij} will be the priority of the element that will be used by the synthesizer.

$$o_{ij} = \sum_k o_{ijk} \quad (5)$$

$$p_{ij} = \sum_k p_{ijk} \quad (6)$$

3. Compute $o_{i\%}$ the total occurrences of elements of all types at level of abstraction < in all matching buildings and $p_{i\%}$ the sum of their priorities. o_i is needed to determine how often an element is used in comparison with the other elements at the same level of abstraction, and *Pi* will be used to normalize the priority of each element at the same level of abstraction.

$$O_i = \sum_j o_{ij} \quad (7)$$

$$p_i = \sum_J p_{ij} \quad (8)$$

4. Normalize the priorities computed in (6) according to the following formula.

$$P_{ij} = 10 \times \frac{p_{ij}}{p_i} \quad (9)$$

After normalization, P_{ij} is a real number between 0 and 10, where 0 indicates the lowest priority and 10 the highest.

5. Select the elements to be used in synthesis. The decision about whether an element *j* should be selected or eliminated is made heuristically according to $o^{\wedge}$, its occurrence, o_{ijt} the occurrences of other elements at the same level of abstraction x, the total occurrence o_t of all elements at the same level, and the number of matching buildings. Table 4-4 shows the decisions made in different cases. Each row in the table corresponds to a production rule in STRUPLE.

Number of Matching Buildings	o_{ij}/o_i	o'_{ij}/o_i	*Disposition*
>-10	- 0	any	eliminated
	<5%	>» 50% (at least one)	eliminated
	<5%	< 50% (all)	selected
	>»5	any	selected
5 - 9	<5%	>» 75% (at least one)	eliminated
	<5%	< 75% (all)	selected
	>- 5%	any	selected
<-4	any	any	selected

Table 4-4: Decisions in Different Cases

In Table 4-4, the first column represents the number of matching buildings. The second column represents the percentage of the occurrence of an element *j* at level *i* in the occurrence of all the elements of level /. The third column represents the percentage of the occurrence of any other element at level *i* in the occurrence of all the elements of the same level. A smaller o_{ij}/o_t indicates less use of element; compared to other elements at the same level. The fourth column in the table indicates the decision about an element's disposition, i.e. whether it is selected or eliminated. All the cases considered fall into three categories according to the number of matching buildings found in the database. The

number of matching buildings reflects how much experience is available. When the number of matching buildings is less than 4, which means that little experience exists concerning the structural systems that could be used for the input building, all the elements of the design space that STRUPLE knows about are selected. When there are a few (5 - 9) matching buildings, which means that some experience is available but not enough to ensure that certain structural elements not used or rarely used in the matching buildings would be inappropriate for the input building, the occurrence of the element against the occurrence of all the others at the same level of abstraction is examined. An element which is not or rarely used is eliminated only when there is at least one other element at the same level of abstraction that is heavily used. Finally, when many matching buildings are found, an element that is not used in the matching building is eliminated at once and those that are not used often are selected only when no other elements at the same level are frequently used. The figures that are used to define different cases were chosen by the authors.

Material Alternatives (Total Matches = 10)			
Type	*Occurrence*	*Priority*	*Disposition*
high strength steel	0.6	5.55	selected
prefab-prestressed re	0.2	2.40	selected
cast-in-place prestressed re	0.1	1.11	selected
composite concrete	0.1	0.94	selected
masonry	0.0	0.0	eliminated
steel	0.0	0.0	eliminated
reinforced concrete	0.0	0.0	eliminated

Table 4-5: A Typical Output of Elements at Material Level

A typical output of the elements at the material level may look as shown in Tabel 4-5. After the disposition of all the elements is determined, it may be necessary to let the user check the design vocabulary selected by the system and delete or add some elements.

4.5. **Implementation**

STRUPLE is implemented in OPS83, a language that combines production and Pascal-like modular programming. The database used in STRUPLE, called *bidgs,* is implemented with Ingres, a relational database management system. The communication between the database and the expert system is set up through a C procedure call. In this section, some implementation details of STRUPLE are given to show how the program is organized and how the different environments and tools are coordinated.

4.5.1. Productions, Procedures and Modules in OPS83

Most of the programming for STRUPLE is done in OPS83. OPS83 has two different program units, production rules and procedures. In STRUPLE, rules are used to represent the knowledge, such as setting up the matching criteria, evaluating matching buildings and selecting design vocabulary. Rules are also used as procedures to perform some computational and output (printing) operations related to the working memory elements (WMEs). Procedures are used for low-level operations, e.g. reading numbers and printing menus and are called from the rules as macro-actions.

OPS83 allows flexible modulization which makes it easy to organize, manage, and modify a medium or large size program. STRUPLE is organized into twelve modules. Each of them either contains rules and procedures and performs a major task or is only a group of procedures and functions that serve as utilities

for other modules.

Instances for criteria are created at the beginning of the program execution and their status are all initialized to no-match. The following rules are typical for determining the status of a criterion and its range/check-list.

Rule: use of above grade stories is a required criterion
IF goal *set_up_criteria La* active
AND criterion *use of above grade stories* has status *no-match*
THEN modify the status of this criterion to be *required*
AND make a check-list item for the criterion containing the same alternative as the use of the current building

Rule: extend check-list of criterion building shape
IF goal *set_upjcriteria La* active
AND criterion *building shape* has status other than *no-match*
AND the shape of the current building is *multi-rectangles*
AND there is no check-list item for this criterion containing alternative *rectangle*
THEN make a check-list item for the criterion containing the alternative *rectangle*

Rule: wind load Is a required criterion
IF goal *set_upjcriteria* is active
AND criterion *wind load* has status *no-match*
AND current above grade stories is greater than 30
THEN modify the status of this criterion to be *required*
AND make a range for the criterion to be from 0.85 to 1.15 times of the wind load applied on the current building

Rule: wind load is a desired criterion
IF goal *set_upjcriteria La* active
AND criterion *wind load* has status *no-match*
AND current above grade stories is no more than 30 but not less than 5
THEN modify the status of this criterion to be *desired*
AND make a range for the criterion to be from 0.85 to 1.15 times of the wind load applied on the current building

4.5.2. Ingres Database

The relations in the database are defined using the same organization as described in Section 4.2. Examples of three typical relations are given below.

GENERAL Rotation						
caseno	***buildingname***	***city***	***state***	use	***unitcost ($psi)***	***totalcost ($)***
5300	Holiday Inn	Greensboro	NC	commercial	24.41	6000000
7000	IBM Office Building	South Field	MI	commercial	67.40	17730000
7400	Elm Park Towers	Worcester	MA	residential	24.24	3733000

Table 4-6: Example of GENERAL Relation

In the **general** relation, *caseno* is the case number of a building. Each building is assigned a unique integer case number and this number is used as the primary key for locating and retrieving the information about a building. In **lateral_3D** and **lateral_2D** relation, *id\s* an integer used as the identifier

LATERAL-3D Relation		
caseno	*id*	*type*
5300	1	2D-orthogonal
7000	1	core
7400	1	2D-orthogonal

Table 4-7: Example of LATERAL-30 Relation

LATERAL-2D Relation					
caseno	*part_of*	*id*	*type*	*material*	*direction*
5300	2D-orthogonal	1	staggered trusses	steel	y
7000	core	1	braced frame	high strength steel	x
7000	core	1	braced frame	high strength steel	y
7400	2D-orthogonal	1	staggered trusses	high strength steel	y

Table 4-8: Example of LATERAL-2D Relation

for a 3D and 2D lateral systems in a building. A building may have more than one 2D lateral system of different types and each of them has a unique ID number. For these relations **caseno** is used as the primary key and id is used as the secondary key. In **lateral_2D** relation, direction is used to indicate the orientation of a frame. For more details, refer to [Zhao 87].

5. Conclusions

A methodology for applying analogical reasoning to structural design has been described. The motivation for this work is to provide a means for capturing and using design experience directly in an expert system so that the performance of the expert system will not be strictly limited by the surface knowledge that can be formalized. The potential for using design experience can be considered in two areas. One area is the advantage of saving relevant experience that a human designer may have forgotten. The other area is the potential for machine learning by recording and using the design experience of the design expert system itself.

In developing STRUPLE, conventional programming methods, expert system techniques and database techniques are all employed. Although STRUPLE works in a way similar to many other expert systems, it has a character that other current expert systems do not have. In addition to the well compiled knowledge, which can be formulated in forms such as rules, STRUPLE uses uncompiled design experience directly as a supplement to the compiled knowledge. Moreover, integrated into a design system, it could gain experience from the structural solutions the system itself produces, thus incorporating a form of doing-learning cycle.

The methodology presented, although it illustrates the concept of using design experience directly, has some limitations that need to be addressed. The method for determining similar situations, i.e. finding the matches in the database, is based on a fixed set of criteria and cannot consider odd or unusual circumstances, while people are very good at modifying their problem solving method to deal with them. A second limitation is that STRUPLE only matches a building at the top level of abstraction, though sometimes it may be desirable or more appropriate to match a building at one specific level and then determine what experience can be extracted from the results of such a partial matching. Another limitation is the inability of the existing method to benefit from the spatial decisions made in previous situations; currently, the methodology provides a means for reasoning about appropriate structural systems and subsystems, not about the number or location of these systems.

As a first effort, STRUPLE is far from being complete. More studies need be made of the structural

design process and design principles in order to reflect them more accurately and make the analogical approach more powerful and useful.

References

[Bethlehem 85] .
Building Case Histories.
Bethlehem Steel Co., 1985.
Unpublished notes.

[Buchanan 84] Buchanan, B. G. and Shortliffe, E. H.
TheMYCINExperimentsoftheStanfordHeuristicProgrammingProject".
Addison-Wesley Publishing Company, Massachusetts, 1984.

[Carbonell 82] Carbonell, J. G.
Towards a Computational Model of Metaphor in Common Sense Reasoning.
Proceedings of the Fourth Annual Meeting of the Cognitive Science Society, 1982.

[Carbonell 83a] Carbonell, J. G., LarWn, J. H. and Reif, F.
Towards a General Scientific Reasoning Engine.
Technical Report CIP No. 445, Department of Computer Science, Carnegie Mellon University, 1983.

[Carbonell 83b] Carbonell, J. G.
Learning by Analogy: Formulating and Generalizing Plans from Past Experience.
Machine Learning: An Artificial Intelligence Approach.
Tioga Publishing Company, Palo Alto, California, 1983.

[Carbonell 86] Carbonell, J.G.
Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition.
Machine Learning: An Artificial Intelligence Approach.
Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

[Davis 81] Davis, R., Austin, H., Carlbom, I., Frawley, B., Pruchnik, P., SnekJerman, R. and Gilreath, A.
The Dipmeter Advisor: Interpretation of Geological Signals.
Seventh International Joint Conference on Artificial Intelligence, 1981.

[Forgy 84] Forgy, C.L.
The OPS83 User's Manual and Report.
Technical Report CMU-CS-84-133, Department of Computer Science, Carnegie Mellon University, May, 1984.

[Harmon 85] Harmon, P. and King, D.
***Expert Systems:** Artificial Intelligence in Business.*
John Wiley & Sons, Inc., 1985.

[Huhns 87] Huhns, M. N. **and** Acosta, R. D.
***Argo: An Analogical** Reasoning System for Solving Design Problems.*
Technical Report MCC-AI/CAD-092-87, Microelectronics and Computer Technology Corporation, April, 1987.

[Ingres 85] Woodfill, J., Siegal, P., Ranstrom, J., Meyer, M. and Epstein, R.
Ingres Version 7 Reference Manual
Electronics Research Laboratory, College of Engineering, University of California at Berkeley, Berkeley, California, 1985.

[Kraft 84] Kraft, A.
XCON: An Expert Configuration System at Digital Equipment Corporation.
The Al Business: The Commercial Uses of Artificial Intelligence.
The MIT Press, Cambridge, Massachussatts, 1984.

[Lin 81] Lin, T.Y. and Stotesbury, S.D.
StructuralConceptsandSystemsforArchitectsandEngineers.
John Wiley & Sons, Inc., 1981.

[Maher 84] Maher, M.L., Sriram, D. and Fenves, S. J.
Tools and Techniques For Knowledge-Based Expert Systems For Engineering Design.
Advances in Engineering Software, 1984.

[Maher 85] Maher, M.L
HI-RISE:AKnowledge-BasedExpertSystemforthePreliminaryStructuralDesignof HighRiseBuildings.
PhD thesis, Department of Civil Engineering, Carnegie Mellon University, January, 1985.

[Marcus 86] Marcus, S., Stout, J. and McDermott, J.
VT:AnExpertElevatorDesignerthatUsesKnowledge-BasedBacktracking.
Technical Report CMU-CS-86-169, Department of Computer Science, Carnegie Mellon University, 1986.

[Mitchell 85] Mitchell, T., Steinberg, L, Shulman, J.
A Knowledge-based Approach To Design.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7(5):502-510, September, 1985.

[Schodek 83] Schodek, D.
Structures.
Prentice-Hall Inc., 1983.

[Winston 81] Winston, P. H. and Horn, B. K. P.
AnIntroductiontoDatabase.
Addison-Wesley Publishing Company, Massachusetts, 1981.

[Zhao 87] Zhao, F.
Application of AI Techniques to Planning Structural System Configurations.
Master's thesis, Department of Civil Engineering, Carnegie Mellon University, March, 1987.