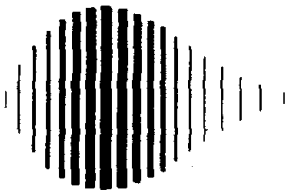
PB91-210930



NATIONAL CENTER FOR EARTHQUAKE
ENGINEERING RESEARCH
State University of New York at Buffalo

A Framework for Customizable Knowledge-Based Expert Systems with an Application to a KBES for Evaluating the Seismic Resistance of Existing Buildings

by

E.G. Ibarra-Anaya and S.J. Fenves
Department of Civil Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Technical Report NCEER-91-0007

April 9, 1991

This research was conducted at Carnegie Mellon University and was partially supported by the National Science Foundation under Grant No. ECE 86-07591.

NOTICE

This report was prepared by Carnegie Mellon University as a result of research sponsored by the National Center for Earthquake Engineering Research (NCEER). Neither NCEER, associates of NCEER, its sponsors, Carnegie Mellon University, nor any person acting on their behalf:

a. makes any warranty, express or implied, with respect to the use of any information, apparatus, method, or process disclosed in this report or that such use may not infringe upon privately owned rights; or

b. assumes any liabilities of whatsoever kind with respect to the use of, or the damage resulting from the use of, any information, apparatus, method or process disclosed in this report.

A Framework for Customizable Knowledge-Based Expert Systems with an Application to a KBES for Evaluating the Seismic Resistance of Existing Buildings

by

E.G. Ibarra-Anaya[1] and S.J. Fenves[2]

April 9, 1991

Technical Report NCEER-91-0007

NCEER Project Number 89-1202a

NSF Master Contract Number ECE 86-07591

1 Graduate Student, Department of Civil Engineering, Carnegie Mellon University
2 Sun Company University Professor of Civil Engineering, Department of Civil Engineering, Carnegie Mellon University

NATIONAL CENTER FOR EARTHQUAKE ENGINEERING RESEARCH
State University of New York at Buffalo
Red Jacket Quadrangle, Buffalo, NY 14261

PREFACE

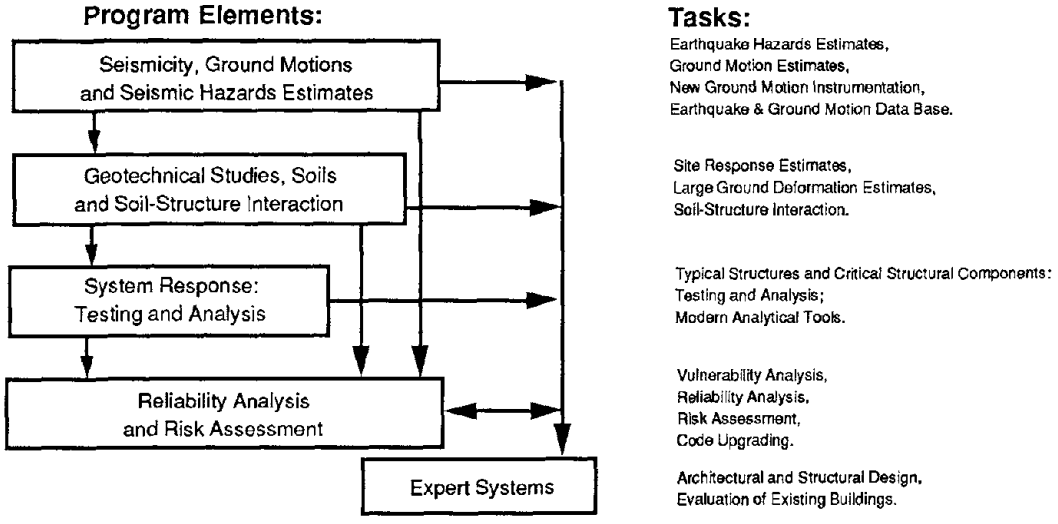
The National Center for Earthquake Engineering Research (NCEER) is devoted to the expansion and dissemination of knowledge about earthquakes, the improvement of earthquake-resistant design, and the implementation of seismic hazard mitigation procedures to minimize loss of lives and property. The emphasis is on structures and lifelines that are found in zones of moderate to high seismicity throughout the United States.

NCEER's research is being carried out in an integrated and coordinated manner following a structured program. The current research program comprises four main areas:

- Existing and New Structures
- Secondary and Protective Systems
- Lifeline Systems
- Disaster Research and Planning

This technical report pertains to Program 1, Existing and New Structures, and more specifically to expert systems development.

The long term goal of research in Existing and New Structures is to develop seismic hazard mitigation procedures through rational probabilistic risk assessment for damage or collapse of structures, mainly existing buildings, in regions of moderate to high seismicity. The work relies on improved definitions of seismicity and site response, experimental and analytical evaluations of systems response, and more accurate assessment of risk factors. This technology will be incorporated in expert systems tools and improved code formats for existing and new structures. Methods of retrofit will also be developed. When this work is completed, it should be possible to characterize and quantify societal impact of seismic risk in various geographical regions and large municipalities. Toward this goal, the program has been divided into five components, as shown in the figure below:



Expert systems constitute one of the important areas of research in Existing and New Structures. Current research activities include the following:

1. Evaluation of existing buildings in terms of seismic safety.
2. Design of new buildings both from the architectural and structural viewpoints.
3. Recommendations for the upgrading of deficient buildings.

The ultimate goal of projects concerned with expert systems is to construct practical expert system tools for the design of new buildings and for the evaluation of existing buildings, with the explicit consideration of expert opinion and uncertainties.

This report describes the development and implementation of a unique capability that will permit users to add customized knowledge to the expert system EVAL for the evaluation of existing buildings for earthquake loads. With this facility, engineering offices can incorporate their custom knowledge to EVAL, in addition to the common knowledge already present. Thus EVAL could contain both the standard knowledge and also the valuable design practice or rules that are specific to an engineering office.

ABSTRACT

The evaluation of existing buildings for seismic resistance is a major technical and economic issue, particularly in sections of the US where existing buildings have been built without explicit consideration for seismic effects. Since many aspects of seismic resistance evaluation and upgrading recommendations are subject to judgement and expertise, the application of knowledge-based expert system (KBES) methodology is warranted. However, it would be impossible to create a KBES with a single knowledge base consistent with the expertise and practice of all professionals performing evaluation and retrofit recommendations.

The report presents an approach towards developing a framework for customizable KBES containing *core* knowledge base commonly agreed-upon procedures and providing means for individual organizations to enter and use their *custom* knowledge reflecting their individual heuristics. The proposed framework is illustrated by means of EVAL, a KBES for the evaluation of seismic resistance of existing buildings based on the proposed framework. EVAL contains only a rudimentary custom knowledge base, but can be further customized by experts using the knowledge acquisition facility provided.

ACKNOWLEDGMENTS

This work was sponsored by the National Center for Earthquake Engineering Research under grant numbers 86-4022, 87-1009, 88-1006A and 89-1202A. The support of the sponsors and the comments of NCEER researchers at Cornell and Lehigh Universities are gratefully acknowledged.

This report is based on the thesis submitted by Enrique Ibarra-Anaya in partial fulfillment of the requirement for the degree of Doctor of Philosophy in Civil Engineering. The work was performed under the supervision of Dr. Steven J. Fenves, Sun Company University Professor of Civil Engineering.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONT'D)

LIST OF ILLUSTRATIONS (CONT'D)

SECTION 1
INTRODUCTION

This section describes the objectives and organization of this work.

1.1 Objective

The principal objectives of this project were:

- the development of a framework for knowledge-based expert systems that allows customization of the knowledge base; and
- the implementation of such framework in the domain of seismic evaluation of existing buildings.

Some expert system applications require the customization of their knowledge bases to accommodate the experience and judgment of different groups of users. Many of the issues in the domain of the application may not have a single *correct* definition, or there may not be a single *correct* procedure to solve these issues. In expert systems dealing with such domains, there should be room for the customization of subjective issues in order to allow groups of users with different opinions to share the same application without conflicts. However, specifically in engineering, not all the tasks to be performed by the expert system will be subject to customization, since there is a large pool of commonly agreed domain knowledge not subject to individual interpretation. This coexistence of subjective and non-subjective knowledge in an application should be identified and taken into account in the design of knowledge-based expert systems.

This work tries to highlight the differences between commonly agreed and custom knowledge in an application and proposes a framework that accommodates both types of knowledge, as well as a facility for modifying the custom knowledge.

The framework developed according to the approach described above was applied to the domain of seismic evaluation of existing buildings. The domain of seismic evaluation of existing buildings was a good candidate due to the following:

- it is a domain that incorporates individual judgment and heuristic knowledge; many aspects of evaluating existing buildings are subject to individual experience;
- at the same time, the seismic evaluation of existing buildings relies on a body of common knowledge, such as structural analysis and structural mechanics; and
- the large number of buildings designed without considerations of seismic loading warrants the development of computer aids for the evaluation of their seismic resistance.

1.2 Organization

The remainder of this work is organized into the following sectionss.

Section 2 **Background**

Previous work in research areas related to the development of this work is reviewed. The first part of this section gives a general description of expert systems and some of the development environments for expert systems. Some methods and tools for knowledge acquisition are reviewed, as well as previous approaches to conflicting knowledge in expert systems. The second part of this section presents the relevant sources for knowledge in the domain of seismic evaluation of existing buildings, and a review of current and previous KBES developed in the domain of earthquake damage mitigation.

Section 3 **A Framework for Customizable Knowledge-Based Expert Systems**

This section presents some common characteristics of certain expert systems applications and gives the rationale for customization. An example of a real engineering application is described to further illustrate the customization needs presented earlier. Finally, a framework for expert systems is proposed that addresses the needs described in the first part of this section.

Section 4 **Knowledge Acquisition Facility**

This section presents the representation scheme selected for the custom knowledge in an expert system application and Eval-KAF, a facility developed for modifying the custom knowledge.

Section 5 **EVAL: A Knowledge-Based System for Evaluating the Seismic Resistance of Existing Buildings**

This section presents the implementation of the framework for expert systems proposed in section 3 to the domain of seismic evaluation of existing buildings. The section is a complete description of EVAL, covering: overall architecture, majors components, control of the system, and evaluation tasks contained in the core knowledge base.

Section 6 **Custom Knowledge Base of EVAL**

This section describes the customization of the core knowledge base.

Section 7 **Examples of EVAL**

This section contains the evaluations produced by EVAL of a sample building and a discussion of the results produced by EVAL.

Section 8 **Conclusions**

This section presents the conclusions and potential extensions to the current work.

SECTION 2
BACKGROUND

This section provides the background relevant to the study, organized into two primary parts. The methodology background addresses the following issues:

- a general description of expert systems and their characteristics;
- some of the existing environments for developing expert systems;
- an introduction to knowledge acquisition and the existing tools for the different types of knowledge acquisition; and
- a survey of knowledge-based expert systems (KBES) approaches to conflicting expertise.

The earthquake engineering background presents the following issues:

- a survey of methodologies for evaluating the seismic resistance of existing buildings; and
- a description of previous or current work on KBES applied to the area of earthquake engineering damage mitigation.

2.1 Methodology Background

2.1.1 Introductory Concepts

There is no standard and widely accepted definition of *expert systems*. Rather, there are different perspectives on what they are and what their defining characteristics are. The two most common perspectives are as follows [12]:

- **Performance perspective.** Expert systems are *"systems that are experts"*, i.e., computer programs that perform competently on tasks that are usually performed by human experts. The tasks are usually limited in depth and breadth, and involve symbolic cognitive reasoning of the sort required in fields that involve specialized training.
- **Methodology perspective.** Expert systems are a commitment to a certain system architecture with characteristics such as: explicitness of knowledge encoding, transparency, availability of explanations and several other features.

A traditional algorithmic application is organized into two parts: data and program. An expert system separates the program into an explicit *knowledge base* describing the problem solving knowledge and a control program or *inference engine* which manipulates the knowledge base. The data portion or *context* describes the problem being solved and the current state of the solution process.

Expert systems have been developed for a variety of tasks. There are two broad classes of tasks for expert systems [11]. The first is the set of *derivation* or *classification* tasks in which solutions are *derived* from given facts and data. This class includes:

- *diagnosis*, identification of the reasons for a failure or malfunction;
- *interpretation*, trying to discern a pattern from a set of facts or readings so that their meaning can be understood; and
- *monitoring*, tracking the changes in a system dynamically with a view towards intervening if something critical is about to happen.

The second class of tasks are the *formation* or synthesis tasks in which a problem is solved by *forming* or creating and object or a set of plans for making an object. This class includes

- *planning*, laying out the sequence of steps required to achieve some objective, and
- *design*, creating an artifact or set out a plan for making that artifact.

The nature of the domain knowledge and the appropriate scheme(s) for representing that knowledge depend in part of the nature of the task to be performed. It is clear that before building a knowledge base that represents the domain knowledge in the application of interest, an adequate framework for modeling the knowledge of the expert in that particular domain must be developed. Some tasks, such as the analysis of X-rays, seem to rely on highly developed pattern recognition skills. Others, such as medical diagnosis, seem to involve primarily specialized knowledge.

A KBES is a computer program that contains the specialized knowledge of an expert or a group of experts in a particular domain and reproduces this knowledge automatically with a predefined problem-solving strategy. In the remainder of this work, the term KBES is used to refer to an expert system.

2.1.2 KBES Development Environments

The four major types of development tools for creating KBES applications are listed below.

- *Programming languages*, such as LISP and PROLOG, may be used to develop entire applications.
- *Domain-independent KBES frameworks*, usually referred to as *shells*, provide a single representation structure and inference mechanism (typically production rules with forward and/or backward chaining, or inference nets) which can be augmented with domain-specific knowledge. These tools have their roots in specific KBES projects, from which they derived their control strategies. Examples of these KBES shells include EMYCIN, KAS, and HEARSAY III.
- *General KBES shells*, such as OPS5 and KES, with built-in inference engines allow the applications developer to concentrate on acquiring the knowledge base and creating rules. The general KBES shells differ from the domain-independent KBES frameworks in that they are not tied to a particular control or inference strategy.
- *Integrated KBES development environments*, such as KEE, Knowledge-Craft, and LOOPS, are programming languages designed specifically for knowledge engineering. They are not

tied to any particular architecture or knowledge representation, but they facilitate the implementation of a wide range of problems. The better environments of this type offer spreadsheet and database interfaces and user interface utilities for application generation, and are implemented on a range of hardware platforms including engineering workstations.

The term *shell* is sometimes used to include general purpose languages (e.g. OPS5) as well as KBES frameworks such as EMYCIN [11]. The word *shell* should be used to describe an KBES framework in which a commitment has been made to a particular knowledge representation and inference strategy. The use of a particular shell requires a good match between that shell's representation and control strategies and the application of interest. General purpose language environments offer greater flexibility because they are not tied to a particular control or inference strategy.

The remainder of this section provides brief descriptions of some of the commercially available KBES building tools. The market is developing very rapidly and no compilation would be completely up to date. For a more complete description of the commercially available environments for KBES, see [22] and [29].

C. Although C is not traditionally an AI language, it has been used to build several KBES shells, such as OPS83, TOPSI and RBC (Rule-Based Calculator). OPS83 and TOPSI are commercially available languages and RBC is a rule-based programming language developed by Skidmore, Owings & Merril for structural design [34].

LISP. Lisp was developed by John McCarthy in the late 1950's as a specialized language for artificial intelligence applications. In Lisp, there is no essential difference between data and programs, hence Lisp programs can use other Lisp programs as data. Lisp is highly recursive, and data and programs are both represented as lists.

Lisp relies on dynamic memory allocation of space for data storage. Unlike other computer languages, such as Pascal or C, memory management in Lisp is completely automatic. The disadvantage of dynamic memory allocation is the need, in some Lisp environments, to stop program execution for significant periods of time to do *garbage collection* when the computer runs out of memory space. *Garbage collection* is the process of reclaiming previously assigned memory space for reuse. This is a drawback for applications in which real-time program execution is important. LISP has been used to create most of the KBES building tools currently being offered for sale.

PROLOG. Prolog was initially developed in 1972 by A. Colmeraruer and P. Roussell at the University of Marseilles. Prolog is a programming language that implements a simplified version of predicate calculus and is thus a true logical language. Prolog has been widely used in Europe, Japan and Australia for AI applications, but less so in the United States.

Like Lisp, Prolog is designed for symbolic rather than simply for numerical computation. Prolog is a higher level language than Lisp in that it has some kinds of deduction and search facilities already built in. Prolog is an interpreted language and thus responds to any query by attempting to return an answer

immediately. A program in Prolog consists of the following:

- specification of facts about objects and relationships;
- specification of rules about objects and relationships;
- queries about objects and relationships.

In a sense, computation in Prolog is simply controlled logical deduction [22]. The user states what he/she knows (the facts) and Prolog is then prepared to tell whether or not any specific conclusion can be deduced from those facts. In knowledge engineering terms, Prolog's control structure is logical inference.

The key idea underlying logic programming is *programming by description* [18]. A Prolog programmer does not specify how the computer is to perform its tasks, but rather a description of the task as a sequence of constraints to be satisfied. In traditional software engineering, a program is built by specifying the operations to be performed in solving a problem, that is, by saying *how* the problem is to be solved. In logic programming, a program is constructed by describing its application area, that is, by saying *what* is true. Said in few words: in LISP one must specify the *how* of a computation, while in Prolog one need only to specify the *what*; it is the task of Prolog to determine the *how*.

Prolog lacks the representation flexibility offered by other knowledge representation schemes, such as frame-based and object-oriented programming. Moreover, there are many kinds of search or program control strategies besides depth-first backward chaining that a knowledge engineer may wish to employ in a KBES. Prolog and particularly Lisp are flexible enough languages in which to build arbitrary search strategies for a given application, but building them can be a very time-consuming and tedious programming job.

KES. Knowledge Engineering Systems (KES) is a product of Software Architecture and Engineering [22]. KES is a family of products that support the development of consultation-style knowledge systems. KES is suited for diagnosis/prescription problems. Facts in KES are represented as attribute-value pairs with associated confidence factors or probabilities. Attributes and values can be arranged in hierarchies. Incomplete or inexact knowledge is represented with probability coefficients or certainty factors (ranging from -1 to 1) attached to the attribute-value pairs. Relations among facts are represented as production rules, using statistical pattern classification techniques.

From the knowledge engineer's perspective, KES is a batch-processing knowledge system. A knowledge base is written with an external editor and stored as a file. Then KES is executed and the knowledge based file is processed. Syntax and other errors appear at run-time. During a consultation session the knowledge engineer can assert values for attributes and thus examine how the system reasons under varying conditions. KES does not supports graphics.

KES is implemented in a variety of Lisp dialects, such as Wisconsin Lisp, Franz Lisp, A-Lisp (for the Apollo workstations) and IQLISP for personal computers.

OPS5. OPS5 (Official Production System 5) was originally developed by Charles Forgy at Carnegie

Mellon University as a tool for psychological research aimed at understanding human memory and cognition [15]. Specifically, OPS5 was developed to test Newell and Simon's hypothesis that *production rules* are sufficient to explain most human cognitive behavior. OPS5 is a knowledge engineering language that supports the rule-based representation model, also known as the *production-system* model.

In production-systems, the basic unit of computation are data-sensitive unordered rules. An OPS5 program is composed of two sections: a declaration section and a production section. The declaration section includes the definitions and descriptions of the data types and the user-defined functions contained in the program. The production section contains the production rules. During execution, the data operated on by the program are kept in *working memory* and the rules are kept in *production memory.*

The basic control mechanism in OPS5 consists of matching, selecting and executing production rules. The inference engine in OPS5 directly supports only forward chaining, but it is possible to implement backward-chaining. OPS5 imposes no constraints on the type of application program that can be written. As a result, the programmer must explicitly handle both data representation and flow of control by means of rules.

OPS5 was originally implemented in Lisp and can be compiled in BLISS to run on Vax-11/780 computers. OPS5 is currently available for variety of hardware platforms. OPS5 is inefficient for solving tasks where an important percentage of the task is represented by quantitative computing. OPS5 supports a single programming paradigm and this characteristic limits its range of applications.

OPS83. OPS83 is a programming language for KBES developed by Production Systems Technologies Inc.. OPS83 is a descendant of OPS5 and it differs from OPS5 in many respects. Some of the unique features of OPS83 are presented below.

- OPS83 supports the procedural as well as the rule-based programming paradigm. This is important because it has become clear that rule-based programming alone is simply not appropriate for some programming tasks. Procedural functions and procedures may be written by the application developer, using a syntax similar to Pascal's, and then called from either the left hand side (LHS) or right hand side (RHS) of any rule. OPS83 offers a comprehensive interface between rules and procedural code.

- The OPS83 system is a compiler, generating object-linkable code that can be combined with code written in a language other than OPS83, such as C or Ada.

- The Rete algorithm [16] for pattern matching is built into the OPS83 run-time system, and the application developer can access the Rete network with predefined functions and procedures. This access is useful when implementing the recognize/act cycle for an OPS83 application, which unlike most other commercially available production-system tools, is not prespecified.

- The run-time performance of a production-system written in OPS83 is five to ten times faster

than the equivalent Lisp-based production system on the same machine [8], providing a significant basis for real-time information processing.

OPS83's most significant disadvantage is that while an application developed in OPS83 is 100% source code portable between machines for which an OPS83 compiler is available, the variety of computers currently supported by Production Systems Technologies (PST), the company that distributes OPS83, is somewhat limited compared to the variety of machines that have C compilers available. Also, PST is the only vendor for the product.

KEE. KEE (Knowledge Engineering Environment) is an integrated knowledge engineering tool for developing KBES implemented in Lisp and developed by IntelliCorp (formerly IntelliGenetics) in the early 1980s to be used specifically in genetic engineering. Later, IntelliCorp promoted KEE as a general-purpose KBES development tool. KEE includes a general-purpose knowledge engineering language that is used primarily for frame-based and object-oriented knowledge representations. It also supports rule-based and procedure-oriented representations.

KEE's basic representational paradigm is *frames*. Facts and rules in KEE are represented as objects or frames that have labeled slots containing either values or means for obtaining values. Slots can contain a number of different entities. A slot may contain a *procedural attachment*, that is, a set of instructions that compute a value for a slot. Similarly, a slot may contain a set of rules that conclude values for other slots in the frame. Procedural knowledge can also be inserted in a slot as a Lisp program. A slot may also point to another frame and indicate an inheritance relationship.

KEE can accommodate multiple knowledge-bases and has an interpreter that can handle both forward and backward chaining mechanisms. It contains a graphics utility used in debugging and interfacing with the user. Using KEE's graphics editor menus, the knowledge engineer can design and construct graphic models of physical objects, meters, and gauges.

Some KEE users complain that not all the features specified in the manual of KEE are possible, and that documentation is not always very accurate.

KnowledgeCraft. KnowledgeCraft is a knowledge engineering tool for building KBES developed by Carnegie Group Inc., based on concepts and methods that originated at Carnegie Mellon University. It is a general-purpose software package that includes a knowledge representation language called Carnegie Representation Language (CRL) and a knowledge base editor. KnowledgeCraft supports frame-based, logic-based, rule-based and object-oriented knowledge representation methods for KBES development. It also supports database system development.

KnowledgeCraft contains utilities that assist with the creation of graphics and menu systems and allows for the use of natural language. It incorporates numerous control mechanisms, including an OPS5 forward-chaining system and a PROLOG-like inferencing system. The CRL-OPS combination provides KnowledgeCraft with a forward-chaining rule-based system, while the CRL-Prolog combination provides a

backward-chaining capability. Both CRL-OPS and CRL-Prolog can be used simultaneously in a single application. KnowledgeCraft is implemented in Common Lisp and can run on many workstations, including some Lisp machines.

LOOPS. LOOPS is a knowledge engineering environment developed at the Xerox Palo Alto Research Center (Xerox PARC). LOOPS is a software tool that incorporates a variety of different knowledge engineering constructs in one unified package. The constructs include object-oriented programming, the use of active values, a knowledge base management scheme, and a rule package.

An active value operates like a probe. By examining an active value, one can see the current status of a variable being reasoned about. By attaching a graphic picture, such as a gauge or thermometer, one can see an analog representation of a variable that is being reasoned about. Moreover, one can monitor changes in that value as processing continues. LOOPS offers the possibility of reviewing programs under development with a variety of graphical schemes.

LOOPS is more an environment than a tool. In order to build an expert system with LOOPS, one must choose from a variety of different approaches and write a fair amount of code before starting to structure the knowledge base. The utility of LOOPS is primarily as a software engineer's environment, where a variety of useful subroutines have been prepared and are ready to assemble.

The development tools for creating KBES presented in this section were considered for implementing the KBES application of this work. The final selection of the development environment is presented in Section 5.

2.1.3 Knowledge Acquisition Methods and Tools

The task of knowledge acquisition for KBES involves building a model of a human expert's knowledge in some specialized (usually fairly narrow) domain, and incorporating that knowledge into an automatic reasoning system which is then able to solve the same type of problems as the human expert. Knowledge acquisition is a major bottleneck in the development of expert systems; it typically involves long discussions between domain experts and knowledge engineers. In addition to extracting the appropriate information, the knowledge engineer must translate the knowledge of the expert into the formal representation scheme used by the KBES. Throughout this process, there must be feedback from the domain expert, and the knowledge engineer repeatedly refines the system until it achieves something close to the expert's level of performance in problem-solving.

The human expert's knowledge is vast and complex, making knowledge acquisition a difficult task. Knowledge acquirers need to be able to understand the expert's often incomplete explanations and be able to structure and categorize the expert's knowledge. They need to have the ability to specify and document the acquired domain knowledge based on a good knowledge of AI paradigms and the possible knowledge representations that may be used.

A wide range of methods and tools, from structured human interviewing techniques to knowledge base

editing tools, has been developed to facilitate the task of knowledge acquisition [5]. The variety of methods in existence reflects the fact that knowledge acquisition is a multidimensional process; it can occur at different stages in the development of an expert system, and can involve many types of knowledge.

Classification of Knowledge Acquisition Methods. According to Hayes-Roth [23], there are five main stages in the development of a typical KBES:

- *Identification.* In this stage one identifies the problem characteristics.
- *Conceptualization.* Knowledge about the concepts, relations and problem-solving strategies is acquired from human experts, textbooks, etc.
- *Formalization.* Conceptual knowledge is translated into structured forms, using a formal knowledge representation paradigm.
- *Implementation.* Rules are formulated that embody the knowledge.
- *Testing.* Performance is evaluated over a large number of test cases.

The available tools and methodologies for knowledge acquisition can be divided into two categories, corresponding roughly to the second and third stages of KBES development described above:

- methodologies and tools for knowledge extraction from human experts; and
- tools for entering knowledge into a knowledge base.

These categories are distinguished mainly by the knowledge source that each methodology uses. In the first category, the knowledge source is a domain expert, and the problem lies in uncovering his/her expert knowledge, and translating it into some intermediate representation understandable by the system builder, typically in the form of natural language documents, diagrams, tables, etc.. Research in this area focuses on the psychology of expert knowledge and problem-solving strategies. In the second category, the knowledge source is usually some intermediate representation of knowledge extracted from domain experts (e.g. as a result of the application of tools in the first category). The problem lies in accurately entering this knowledge into the knowledge base of the KBES. Tools to aid in this process are usually designed for the system builder, and focus on knowledge representation issues for expert knowledge, information presentation techniques, and user interface technology.

Human Interviewing Methods. The two main approaches to extracting knowledge from a human expert for the purpose of constructing a KBES are [10]:

- rapid prototyping; and
- structured knowledge acquisition.

In the rapid prototyping approach, the knowledge engineer conducts a minimal amount of human interviewing and then quickly implements a prototype system; the prototype is then successively refined as further rounds of interviewing determine gaps and errors in the system. The second approach,

structured knowledge acquisition, defers implementation and instead maps the knowledge of the expert into an intermediate level representation. It is desirable to maintain independence from the final knowledge representation language during knowledge acquisition so that the expert can express his/her knowledge in as natural and direct a representation as possible.

Grover [21] proposed a formal methodology for extracting, recording and organizing human expertise into a hand-written knowledge acquisition document series which multiple experts and designers can access and update.

The previously described techniques for knowledge acquisition require a human intermediary (a knowledge engineer) for performing knowledge extraction from domain experts. Prototype knowledge acquisition systems that interview experts have been built in several KBES problem areas. These automatic interviewing systems help to eliminate some of the inherent problems of human interviewing, such as bias and inconsistency, particularly when there are multiple persons interviewing multiple domain experts. Two of these automatic interviewing systems are reviewed below.

ETS. The Expertise Transfer System (ETS) is a tool developed by John Boose that interviews experts to help them build KBES [7]. The interviewing methodology is based on techniques from personal construct theory used in psychotherapy. ETS uses a psychological model of human behavior proposed by Kelly [30] in 1955, and applies some empirically tested interviewing techniques. Kelly theorized that each individual behaves as a *personal scientist*, seeking to predict and control the environment by forming theories and hypotheses, and testing them by categorizing and classifying experiences. Kelly assumes that there is fundamentally no difference between and individual supporting his *point of view* and a scientist supporting his theories.

Kelly developed certain techniques for use in psychotherapy based on this philosophy. The techniques assume that individuals are capable of expressing their conceptual structure as a series of *bipolar dimensions* that Kelly called *constructs*. It is through this system of personal constructs that an individual constructs events and categorizes experiences. Since different individuals perceive the world with different construct systems, individuals are capable of making a unique interpretation of the same event. Kelly named his collection of ideas about this process *personal construct theory*.

Kelly felt that patients who were having trouble with the world probably had some major inconsistencies in their network of constructs. He developed techniques that allowed individual patients to examine their own systems of constructs, thereby helping them verbalize and structure their attempts to predict, control and understand the world. Psychologists use personal construct methods to help patients explore their views of the world; ETS uses these methods to help experts explore the way they solve problems.

Kelly developed several methods for eliciting and analyzing personal construct systems. His most well-developed method, the Role Construct Repertory Grid Test, is used in a modified form by ETS. ETS employs an automated implementation of the repertory grid test, to allow human experts to identify and classify their problem-solving knowledge. For a more in-depth look at personal constructs and the

repertory grid test, see [7], [30], [32] and [31].

ETS proceeds through several phases of interaction with the domain expert in order to acquire a classification of domain concepts/goals and parameters; from this, ETS can derive implication rules. The output of ETS can be automatically translated into production rules in several knowledge representation languages: KS-300/EMICYN, KEE, SUDV (a Boeing C-based tool), MRS (a research tool from Stanford University) OPS5, LOOPS, Prolog, M.1 and S.1.

The ETS approach suggests promising directions for future research in automating the knowledge acquisition process. However, it has several limitations. The *implication rules* extracted from the rating grid only indicate correlations in the data, and do not necessarily reflect underlying causal relations. It is difficult to apply the repertory grid methodology to elicit causal or procedural knowledge. ETS elicits traits and builds relationships but does not find out much about how this information is used in the problem-solving process. Kelly's grid methodology is best suited for derivation problems (e.g., diagnosis and interpretation) and is not readily applicable to formation problems (e.g., planning or design), or problems that require a combination of analysis and synthesis. Another problem with ETS is that it is designed specifically for KBES which use the production rule representation. Consequently, ETS attempts to elicit a mixture of knowledge about concepts, terminology and problem-solving goals, all in on stage, using a single representation; this may be a confusing and unnatural way for the domain expert to describe his/her knowledge. Finally, the knowledge acquired by ETS does not necessarily constitute a sufficient representation of the problem conclusion set, and errors of omission may be quite difficult for the expert to detect [7].

Thus ETS, in its present state, is best suited as a preliminary knowledge acquisition tool, and must be augmented with human interviewing to fill in gaps in the knowledge base.

Roget. Roget, developed by Benett, attempts to elicit the vocabulary or terminological framework for a diagnostic problem from a human expert [6]. The approach used in Roget is quite different from that of ETS. Roget incorporates a generic conceptual structure for diagnostic tasks consisting of abstract categories of goals, subgoals, advice (or results), evidence (or data), and inference steps. This conceptual structure is derived from Roget's knowledge about several examples of existing diagnostic KBES. The expert initially enters a natural language description of a new task, which Roget analyzes, using rudimentary natural language understanding techniques, to determine the task type. Possible task types in Roget's vocabulary include *determine-causes*, *evaluate-evidence*, etc.. Next, Roget asks what sort of supporting inferences, evidence and data will be used to achieve the goal of the particular task. Roget can tell the expert what general evidence categories it knows about, such as *predisposing-factors* or *laboratory-tests*. It can give domain-specific examples from existing KBES that it knows about. The final result is a graph of domain-specific instances of Roget's generic conceptual structures.

Roget offers a structured environment for acquiring knowledge from experts and its approach appears promising as a way of solving some of the problems of acquiring knowledge from humans. On the other hand, the types of knowledge that can be expressed using Roget are limited to what has been expressed

in the existing diagnostic KBES that Roget knows about. New features and task types cannot be easily incorporated. Roget also has the same limitations in knowledge representation as those discussed for ETS.

Tools for Entering Knowledge into a Knowledge Base. In the remainder of this section, some of the tools that belong to the second category of knowledge acquisition methods (tools for entering knowledge into a knowledge base) will be presented. Some tools in this category are general systems that allow the user to construct a knowledge base from beginning to end, and others are intended specifically for refining existing knowledge bases.

KAS. The knowledge acquisition system (KAS) was developed by Reboh for the PROSPECTOR KBES [40]. KAS is intended to be used by a computer scientist, or someone who is capable of learning and understanding in detail the mechanisms of the computer program, command syntax, etc. KAS provides a complex command language that allows the user to create, modify or delete nodes in semantic, rule and taxonomic networks. This system represents knowledge in two ways: as probabilistic inference rules and partitioned semantic networks. KAS's main limitations are that it requires detailed knowledge about the knowledge base, as well as about the KAS command language which is very complex and cryptic; thus, it is recommended only for computer scientists.

MORE. MORE is a tool developed at Carnegie Mellon University that assists in eliciting knowledge from domain experts [35]. MORE stores domain facts in an event model that represents the relations among occurrences of events and the conditions that affect the relations. From this event model, it generates a rule model of condition/conclusion assertions used to perform diagnosis. Because MORE understands how knowledge is used in diagnosis, it understands what knowledge it needs; it has strategies for detecting errors and for seeking information that will increase the diagnostic power of the knowledge base. MORE's use has been explored in several application areas: diagnosis of drilling-fluid problems, epileptic seizures, disk faults and manufacturing defects in the wave soldering of circuit boards.

SALT. SALT is a knowledge acquisition tool developed at Carnegie Mellon University [36]. SALT is one of the first automated knowledge acquisition methods to address synthesis rather than analysis problems. SALT is intended for use by domain experts to create and maintain systems that perform constraint-satisfaction tasks such as designing an artifact or constructing a schedule. It acquires knowledge from an expert and generates a domain-specific knowledge base compiled into rules. It then combines this compiled knowledge base with a problem-solving shell to create an expert system. SALT uses a propose-and-revise problem-solving strategy. The SALT-assumed method constructs a design incrementally by proposing values for design parameters, identifying constraints on design parameters as the design develops, and revising decisions in response to constraint violations in the proposal.

TEIRESIAS. TEIRESIAS is a knowledge base refinement tool developed by Davis [9]. TEIRESIAS was designed to allow a user to update and debug a MYCIN knowledge base. It provides means for adding new conceptual primitives, attributes and rules. The interaction dialogue is conducted in a restricted subset of natural language. TEIRESIAS's questions are formed mainly by filling in templates

and the user's responses are understood by keyword recognition. TEIRESIAS incorporates two kinds of meta-level knowledge about its object representations: data structures and rule models. This meta-knowledge is used to guide the knowledge acquisition process.

The data structure used in TEIRESIAS is a frame-like representation of generic concepts. By instantiating schemas, the user can add new generic concepts. New rules can be added similarly by instantiating rule models. One limitation of TEIRESIAS is that it assumes that the present content of the rule base is correct, and is a good basis for predicting the structure of new rules; when these assumptions are false, the system sometimes will persist along the wrong path in guiding the user. Another limitation of TEIRESIAS is its *simple-minded* technique for natural language understanding, which presents a substantial barrier to better performance [9]. TEIRESIAS relies on knowing about what is already in the knowledge base to provide help to its user(s); thus, it is only suitable as a refinement tool, and cannot be used as a complete system building tool.

PLANEX. PLANEX is a domain independent knowledge-based process planning system architecture developed at Carnegie Mellon University [47]. Starting from a description of the physical artifact to be constructed or manufactured, PLANEX generates the set of activities needed to create the artifact. The knowledge base of PLANEX consists of the knowledge sources that store the knowledge to generate a plan. Two versions of PLANEX have been developed: CONSTRUCTION PLANEX and HARNESS PLANEX. CONSTRUCTION PLANEX generates activity plans for the excavation and structural erection of concrete or steel-frame buildings. HARNESS PLANEX generates activity plans for manufacturing automotive electrical harnesses.

The system architecture of PLANEX contains a facility for acquiring and updating the required domain-specific knowledge called *KSAM: Knowledge Source Acquisition Module*. KSAM is an interactive editor that lets the user create, modify or delete knowledge sources. After changes are made to the knowledge sources, KSAM translates the tabular representation used to describe the knowledge sources into the appropriate schema representation used by the knowledge source evaluator of PLANEX.

Several other tools for knowledge acquisition exist currently; it is out of the scope of this work to try to present them all. Knowledge acquisition research is in progress in industrial and university laboratories. Effective knowledge acquisition is the key to the realization of useful, cost-effective KBES. KBES are useful only after the requisite subject domain knowledge is acquired, codified and applied.

The knowledge acquisition methods and tools presented in this section served as a basis for the selection of the custom knowledge representation presented later in this work and provided useful guidelines for the design of the knowledge acquisition facility developed.

2.1.4 KBES Approaches to Conflicting Expertise

Experts in the same domain who try to solve the same problem may use different techniques. For instance, two experts may use *different* methods to arrive at *similar* solutions or at *different* solutions, both of which may be acceptable. Experts can also use *similar* methods to arrive at *similar* solutions or even *different* solutions. These differences in techniques and solutions tend to make it difficult to use several experts to build a KBES. Differing methods and solutions will tend to be based on different models of expertise, and the KBES must somehow reconcile these different models and still give reasonable recommendations.

Typically, a KBES will first be built using the knowledge of only one expert. After this initial prototype KBES is built, it is often desirable to add knowledge from other experts in the same problem domain. Difficulties will result from differences in vocabulary, definitions, heuristics, and problem-solving strategies. Several issues must be addressed by the knowledge engineer when combining knowledge from several experts:

- How can experts reach an agreement about vocabulary and definitions ?
- Do they agree on heuristics and solutions ?
- Should the KBES try to model a consensus among a given set of experts when recommending solutions ?
- Should the KBES accommodate conflicting expertise of the possible different users ?

There is currently little or no methodology for combining multiple expertise in the same area to build KBES. Boose [7] conducted several interesting experiments with ETS, using personal constructs and rating grids, described earlier in this section, to combine expertise from many sources. One of the experiments consisted in asking several experts to help build a succession of AI Tool Advisor KBES that would interview an end user and recommend a set of AI tools selected from a larger set. ETS was used to elicit the information and perform negotiations.

The approach proposed by Boose is shown in Figure 2-1. Several experts in the same problem domain are interviewed by ETS to produce independent rating grids that capture their knowledge. ETS combines their knowledge into a single consultation KBES. The consultation system is run on a number of test cases and compared with each expert's expectations. Rank correlation scores are used to measure system performance and the areas of difference between experts are derived. The experts participate in structured negotiation and use ETS to re-rate the rating grids. The performance of the resulting system is again tested and measured.

Four different scenarios were used in the experiments described by Boose to combine expertise from different domain experts. In the first scenario, two experts independently used ETS to build an AI tool

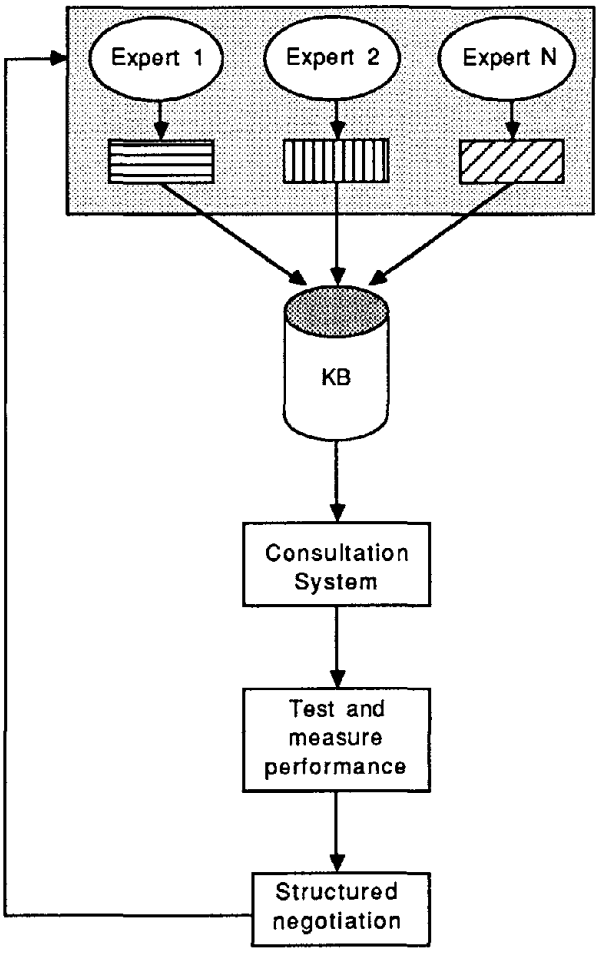


Figure 2-1: Methodology proposed by Boose [7] for interviewing multiple experts

advisor. A different set of tools (elements[1]) and traits[2] was produced by each of the experts. A combined consultation system was then produced by ETS using the rating grids produced by each expert. A coefficient of rank correlation was used to compare ETS's results with the expectations of the two experts.

In the second scenario, a common grid was constructed from the tools and traits of several experts, and then each expert made ratings using the same grid independently. Each of the experts was asked to place his/her own ratings on this new common grid. Some experts had to *guess* what was meant by a trait elicited from a different expert.

[1] *Elements* are the specific conclusion items that will be recommended by the KBES.

[2] *Traits* (also referred to as *attributes* or *constructs*) refer to those aspects of the problem domain that an expert uses to solve a given problem.

In the third scenario, smaller rating grids were used to produce a consultation system using the expertise of three experts. Next, ETS used an algorithm based on MINUS [42] to show the differences between each expert's grid and every other grid. A MINUS grid is produced by taking the difference in ratings between two existing grids with the same elements and traits. These MINUS grids were used to show the areas of most agreement and most difference. This information guided manual negotiations and discussions between the experts. During these negotiations, each of the experts explained what they had meant when using the trait to rate the different AI tools. The experts found that there were significant differences in the meanings attached to the labels, and significantly different ways in which those meanings were used to rate the tools. The experts felt that the negotiation session was fruitful and that they had learned something from discussing the issues.

The fourth and scenario was the simultaneous use of ETS by more than one expert. In this last scenario, two experts used ETS to build a small consultation system. Both experts together picked the recommended tool set (elements), discussed each comparison, generated appropriate traits, and rated each trait against each tool. Again, ETS built a consultation system from this grid. The degree of conflict with this approach was very low and the system's performance was very good, significantly agreeing with the experts' expectations in most cases. In this case, only two experts and a relatively small problem was used. However, this approach might become unwieldy with more experts or with larger problems.

The experiments made using personal construct theory and the knowledge acquisition tool ETS show some interesting approaches for dealing with knowledge acquisition from multiple experts. This approach tries to formalize the process of combining multiple expertise and to locate the areas of agreement and disagreement between the different experts. The approach described by Boose using ETS is certainly one of the best documented frameworks for dealing with multiple expertise. The disadvantages of this method are that the type of problems that can be modeled with this approach is limited by the suitability of using personal construct theory for modeling the domain knowledge of the problem of interest. Rating grids are best suited for representing knowledge in KBES of the *derivation* type and cannot be readily applied to *synthesis* problems.

The experiments performed by Boose are a good and structured approach to combine the individual knowledge of multiple experts. However, as it will be described in the next section, experts will not always agree on the criteria to be embedded in a KBES and consensus might not always be achieved. The following section presents a framework to accommodate conflicting knowledge in which each individual expert is able to express his/her individual knowledge without being forced to try to achieve consensus with other experts in the field.

2.2 Earthquake Engineering Background

Earthquake engineering has been developing at a steadily increasing rate. Research results from many universities and research centers, lessons learned from studies of ground motion records and earthquake effects on structures, and other technological developments have been continuously adopted into practice and incorporated into new codes and standards. Although recent improvements in seismic design codes

and standards have resulted in increased earthquake resistance for new buildings, they have created a dilemma for existing ones.

2.2.1 Evaluation Methodologies

In the past years, several studies have been directed to the seismic safety assessment of existing structures. There are many evaluation methods currently in use for assessing the seismic performance of buildings. They range from very approximate screening methods for identifying potential structural problems in large groups of buildings, to detailed analytical approaches for studying performance of structural members, components, and systems on an individual structure basis. Evaluation methodologies may be classified into several categories:

- screening methods for quick evaluation of large groups of buildings;
- rapid evaluation procedures for individual buildings;
- procedures for more accurate evaluation of individual groups of buildings; and
- detailed analytical approaches, usually involving dynamic analysis and computer-based methods.

There is such a large number of approaches that providing a description of each is out of the scope of this document. In the remaining part of this section, brief descriptions of some of those studies are presented.

ATC Provisions. The ATC (Applied Technology Council) is a non-profit organization created in 1971 to assist structural engineers in the task of keeping abreast and effectively utilizing technological developments.

In 1972, ATC, sponsored by the National Science Foundation and in cooperation with the National Bureau of Standards, initiated a project aimed at the development of practical design provisions. The results were published in 1978 under the title *"Tentative Provisions for the Development of Seismic Regulations for Buildings, ATC-3"* [3]. Although this publication is intended for the design of new buildings, it includes three chapters dealing with the following topics about existing buildings:

- systematic abatement of seismic hazards in existing buildings;
- guidelines for repair and strengthening of existing buildings; and
- guidelines for emergency post-earthquake inspection and evaluation of earthquake damage in buildings.

The relevance of this study is that it includes a chapter on seismic evaluation of existing buildings.

ABK methodology. In 1984, the ABK Joint Venture completed a research project to develop a methodology for evaluating unreinforced masonry buildings in regions of high seismicity [1]. ABK is a joint venture of the Los Angeles structural engineering firms of Agbabian Associates, S.B. Barnes and

Associates, and Kariotis and Associates. This project included:

- a nationwide survey of unreinforced masonry buildings for purposes of categorization;
- quasi-static and dynamic testing of diaphragm, wall, and anchorage elements typically employed in unreinforced masonry; and
- the development of a recommended methodology for seismic evaluation of such structures.

The ABK methodology is recommended for seismic evaluation of unreinforced masonry buildings in regions of moderate or high seismicity. The recommended methodology is an effective tool for evaluating buildings that have essentially flat and solid wood diaphragms with masonry or concrete shear walls around the perimeter of the building and sufficient interior cross walls. This methodology is not recommended for buildings with reinforced concrete floors and/or roofs.

ATC-14. In 1983 ATC initiated a project aimed at developing a procedure for identifying and evaluating seismic hazard in existing buildings. It was originally entitled *"Methods for Evaluating the Seismic Strength of Existing Buildings"*. The findings of the project were published in 1987 under the title *"Evaluating the Seismic Resistance of Existing Buildings ATC-14"* [2].

The overall objective of the ATC-14 project was to develop a comprehensive but practical methodology that could guide engineers in all seismic zones of the United States in evaluating existing buildings to determine potential earthquake hazards and identify buildings or building components that present unacceptable risk to human life.

The methodology is intended to be used as a guide by structural engineers experienced is seismic design and analysis of buildings. A major portion of the methodology is dedicated to directing the evaluating engineer on how to determine if there are any weak links in the structure that could precipitate structural or component failures. Potential weak links have been identified from detailed reviews of building performance data from past earthquakes.

There are two steps prior to the evaluation itself. The initial step is the collection of all documentation of the structure of interest, that is, drawings, specifications, calculations and other information about design, construction, maintenance, and additions or modifications to the structure. Moreover, site visits are also essential for the success of the analysis.

The second step is the classification of the structure of interest. This is very important since the entire evaluation process depends on this classification. The building classification employed in the ATC-14 methodology is based on the material or type of construction used in the principal gravity and lateral force resisting systems.

There is such a large combination of materials and structural systems that to include all of them in the evaluation methodology was not possible. However, considering the construction types which for economic reasons are of prevalent use, 98 different building types were identified. Afterwards, based on

the fact that structures that use like construction materials have exhibited similar performance in past earthquakes, the number of building types was narrowed to 14 different building types considered in the ATC-14 methodology. To facilitate the analysis procedure, those 14 types were grouped into six generic building types, which are: cast-in-place concrete, steel frame, precast concrete, reinforced masonry, unreinforced masonry and wood frame.

The classification of the structure in analysis is important because the ATC-14 methodology is intended to cover building features and the characteristics of each subtype of structure which may present a potential hazard. The evaluation methodology for each subtype consists of the following:

- a general description of the structural features of the specific building type;
- a description of the observed performance of this building type in past earthquakes with specific examples;
- an explanation of the probable load paths for vertical and lateral loads in the structure; and
- a list of statements of concern about the building features which might present a hazard.

2.2.2 KBES Applications

The problem of assessing the seismic safety of a structure is largely heuristic. The nature of this problem makes it a suitable task for the application of KBES methodology. This section is devoted to the literature review of previous or current work on KBES applied to the area of earthquake damage mitigation. KBES in this particular domain can be categorized into three main groups:

- systems that perform evaluation after earthquake damage has occurred;
- systems that provide advice for the design of new buildings; and
- systems that evaluate the seismic safety of existing buildings.

SPERIL I. Speril-I is a KBES developed at Purdue University [28] for estimating the level of damage suffered by a structure after an earthquake, so that repair plans can be made accordingly. The primary concern in Speril-I was the development of its inexact inferencing method.

SPERIL II. Speril-II is an advanced version of Speril-I developed by a joint venture between Purdue University and the firm of Wiss, Janney, Elstner and Associates. Speril-II is based on a number of different testing techniques, along with varying approaches to interpreting visible damage. It is aimed to aid the engineer in deciding whether or not a structure should be repaired or, depending on the level of damage, destroyed.

Speril II's performance has been improved with respect to Speril I by incorporating both forward and backward chaining strategies. In addition, the input observations on local damage are changed into global implications, dependent on the type of component in analysis. In this way, components, such as beams, columns and joists, are described by a-kind-of relations. Consequently, the damage on specific components is related to the structure as a whole. Speril II also combines individual inexact inferences

into local, global and cumulative damage assessments of the entire structure.

When comparing the conclusion reached by Speril II with those of a human expert, the structures are categorized into three classes: satisfactory, questionable and unsatisfactory. These categories indicate the extent to which the KBES matches the opinion of the expert. Further details are provided in [28] and [17].

CMU prototype. A prototype KBES for evaluating the damage state of a structure that has undergone seismic damage was developed at Carnegie Mellon University [43]. The CMU prototype was intended to provide an estimate of the amount of damage that a structure has sustained based on available damage observations. The goal is to make recommendations as to the serviceability of the structure based on the estimate of the extent of the damage incurred during the earthquake. This study was also intended to provide the framework for the current work by investigating appropriate KBES languages and shells and experimenting with appropriate inexact inference mechanisms.

ARCHQUAKE. ARCHQUAKE is a KBES being currently developed at Lehigh University [46]. The primary goal of ARCHQUAKE is to introduce structural design knowledge early in the architectural design phase. ARCHQUAKE is intended to:

- provide a structural design perspective to the architect early in architectural design;
- consider the architectural perspective when rendering structural advice; and
- tailor that advice by knowing something about the intent of the architect.

STRAKE. STRAKE is a KBES being currently developed at Cornell University [46]. STRAKE is being built to assist structural engineers, by checking their design concepts against a knowledge base of facts and heuristics pertaining to earthquake-resistant design.

ASALVO. ASALVO is a KBES developed at MIT as a decision support tool for structural engineers evaluating the seismic resistance of existing structures [38]. ASALVO uses ATC-14 as its main source of knowledge. This system does not make a final decision regarding the structure's overall earthquake resistance; rather, it issues warnings and recommendations to the user if the existence of hazardous features are confirmed. The reason for ASALVO not getting to a conclusive decision is that the knowledge used does not allow such a conclusion. The source of knowledge for ASALVO, ATC-14, is based on a practical, consensus-based methodology of evaluation. However, it has been pointed out by the professionals involved in the development of ATC-14 that the evaluation of a given structure is highly dependent on the judgment of the evaluating engineer. Also, the evaluation depends on the particular characteristics of the structure under analysis. Moreover, in most warning statements, the methodology recommends lengthy calculations in order to arrive at a more definitive conclusion about the seismic resistance of the structure. Hence, ASALVO is only a decision support tool.

AMADEUS. AMADEUS, which stands for the Assessment of Damage after Earthquakes and Usability Structures, is a knowledge based system developed at MIT [44]. It is an advisory system for the

assessment of a building damaged by an earthquake. Based on an inspector's observations, AMADEUS is aimed at helping the engineer to make quick and accurate decisions regarding the degree of building damage and, consequently, its habitability state.

Of the applications presented above, ASALVO is the only system that is intended to evaluate the seismic resistance of existing buildings. The other applications described are intended to assist an engineer in the design of new structures or in the evaluation of structures that have undergone seismic excitation and may be damaged after the earthquake. However, ASALVO is a mere implementation of the rules contained in the ATC-14 methodology and does not perform any type of quantitative analysis of the building.

SECTION 3
A FRAMEWORK FOR CUSTOMIZABLE KNOWLEDGE-BASED EXPERT SYSTEMS

This section presents some common characteristics of a class of KBES applications. These characteristics give rise to the need for customization and the elimination of duplication of effort among organizations developing KBES for similar applications. To further illustrate the customization needs presented, a system developed for a real engineering application is described. Finally, a framework for KBES is proposed that addresses the needs described in the first section of this section.

3.1 Rationale

Knowledge-based expert systems (KBES) provide a methodology for capturing, organizing and utilizing heuristic knowledge accumulated over many years of experience within an organization. A KBES incorporating an organization's expertise can faithfully reproduce the heuristics and problem-solving style that presumably differentiates that organization from other organizations engaged in similar endeavors. To be useful to an organization, a KBES needs to be highly idiosyncratic in handling situations where there is no general consensus about the *correct* way of solving a problem.

A common problem in developing such highly tuned, individualized expert systems is that in order to incorporate the unique specialized knowledge of an organization, the developers also have to incorporate large amounts of knowledge that is common to an entire discipline and not subject to individual interpretation. As a consequence, there may be a great deal of duplication of effort among organizations developing KBES for a particular domain. Worse yet, organizations may abandon or curtail KBES development before they reach the stage where their *custom* knowledge can be introduced. The problem of having to incorporate large amounts of common knowledge further compounds the severe difficulty of knowledge acquisition, acknowledged to be the major bottleneck of KBES development. Another problem that may arise when different organizations develop separately a KBES for the same domain is that some of these organizations may misinterpret or incorrectly encode some portions of the general knowledge of the domain. This possible misinterpretation of the domain concepts and/or terminology may cause confusion among the different professionals involved in the same discipline.

The need for customization in KBES is evident in domains where no single interpretation of the domain concepts satisfies the criteria of all the possible organizations. In such domains, the ability to customize the knowledge base is imperative to make the KBES usable by more than one organization. As described in the last section of the previous section, it is likely that experts may disagree on particular issues in their domain, creating a conflict of expertise. Managing conflict and achieving consensus among multiple sources of true expertise is a difficult task. Disagreement between experts is not inherently bad; it may simply represent different and valid approaches to the same problem. Hence, KBES must accommodate the individual knowledge bases of the different users of the system, if these users have or may have conflicting expertise on specific issues of the domain to be represented in the knowledge base.

The ability of customizing the knowledge base of a KBES, due to conflicting expertise in selected issues of the domain, implies the need of updating the individual knowledge bases of the different users. It is unlikely that the knowledge or heuristics that represent the expertise of an organization or individual user will remain static. Additional experience gained by an individual user or organization and external influences, such as new legislation or modifications of relevant codes, may change with time from the original knowledge specified by the user organization when the KBES was originally developed. For a KBES to be valuable and useful to any one organization, it must be able to grow with newly acquired, highly individual, expertise.

In conclusion, there is a need to separate commonly agreed knowledge from custom knowledge in many KBES applications. This separation requires a very high level conceptualization of the entire domain knowledge to be represented in the knowledge base. According to the different stages of development of a KBES proposed by Hayes [23] and described in the previous section, this separation of the domain knowledge must be made in the conceptualization-formalization stage of development.

3.2 An Example of an Engineering Problem that Requires Customization

There are many engineering problems that require the customization characteristics described in the previous section. To further illustrate these customization needs, this section presents a computer system developed in 1985 by the first author while working for an engineering firm in Mexico City.

A computer program was to be developed for performing analysis and structural design of reinforced concrete retaining walls. The general shape of a reinforced concrete retaining wall is illustrated in Figure 3-1.

The program was to receive as input the following data:

- an initial size and shape of the retaining wall and the portions of soil profile at both sides of the wall;
- the pressures exerted by the soil to the retaining wall;
- the density of the soil contained by the wall;
- the *f'c* and *fy* parameters of the concrete used in the retaining wall;
- the allowable vertical pressure to be resisted by the underlying soil;
- the friction coefficient of the soil; and
- external forces that could optionally be applied to the wall.

The following constraints must be met by the design of the wall and had to be checked by the program:

- the wall must be stable with respect to overturning;
- the friction force generated at the base of the wall should be greater by a given safety factor than the force exerted by the soil to the wall to prevent sliding of the wall;

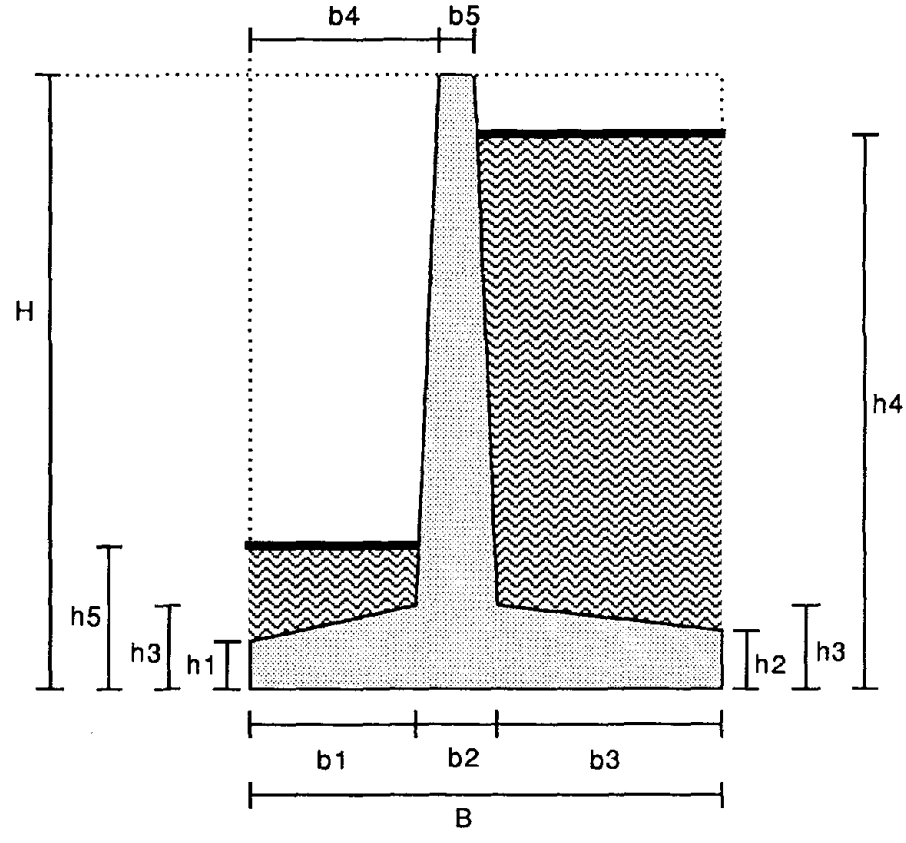


Figure 3-1: Reinforced Concrete Retaining Wall

- the tip deflection of the stem of the wall should be less than a specified limit;
- the pressure developed at the base of the wall should be less than the allowable vertical pressure on soil.

One of the functional requirements of the program was that if any of the above constraints is violated, or if the maximum percentage of steel allowed by the code was exceeded in any component of the wall, the program was to modify the initial size/shape of the wall to try to satisfy violated constraints. If, after a number of modifications to the original design, all of the constraints were not yet satisfied, the program was to abort informing the user what problems there were with the design originally specified.

This program was developed to be used by different groups of engineers working on different projects. One of the groups was working on the design of a new highway, another on the design of the intersection of two freeways, and the last on the design of a box-section tunnel for a subway. These engineers were both the experts who suggested the tasks to be performed by the computer program and the end users of the program. All of the experts agreed on the constraints that the design had to satisfy. Similarly, the experts agreed on the analytical procedures to be followed. However, there were significant conflicts

among the experts about the measures to be taken in case that a certain constraint was violated.

This lack of consensus was a significant problem in the design of the computer program, since the program was to be designed as a single application for all the engineers in the company. It became clear that a single hard-coded solution was not going to satisfy the criteria of all the individual users. If the functional requirements of the program had been purely the analysis portion, no conflicts would have been arisen, since the users agreed on the constraints to be checked and in the analytical procedures to be used. However, it was decided that in order to make the computer program more useful, the program would modify the original design in cases where it was not appropriate.

The solution that was finally taken was the simultaneous generation of multiple designs in cases where a constraint with *non-consensus* corrective measures was violated. Each engineer would then pick the design that resembles most his/her group's criteria, or would refine its original input manually and re-run the program until he/she was finally satisfied with the design. The program worked at the end but its output never truly satisfied the criteria of all the users.

If any of the constraints is violated, there are a number of different corrective measures that can be taken to try to comply with that constraint. Figures 3-2 and 3-3 illustrate some possible measures that can be taken to try to comply with a given constraint.

If a wall tends to slide, there are basically two measures that can be taken to overcome this problem (see Figure 3-2):

- add a *key* at the base of the wall;
- increase the weight of the wall.

Some engineers definitely disliked the use of a key while others considered it an efficient corrective measure. According to the relevant literature in the domain, the addition of a key is a valid measure for increasing the stability of a retaining wall against sliding; however, there are engineers who have their reasons for not liking this option. Among the group of experts that preferred the second option (increasing the weight of the wall) there were conflicts in the strategies to be taken when changing the dimensions of the wall. The program makes several attempts to re-size/re-shape the wall in order to increase the weight of the wall. However, there are multiple options and strategies that can be taken in every successive re-sizing operation. There was significant conflict in trying to define a commonly-agreed criterion. Some engineers preferred to increase the base of the wall first and after several attempts to increase the vertical dimensions. Other engineers disagreed with this strategy and some of them did not accept a significant increase in the size of the base of the wall since excavation in the particular conditions on the project was very expensive. The resizing strategy that was finally selected was the result of negotiations between the engineers but it did not satisfy all the users.

A similar problem occurred when the elements of a wall were subject to excessive internal forces or experienced deflections beyond an acceptable limit. As seen in Figure 3-3, there are multiple alternatives

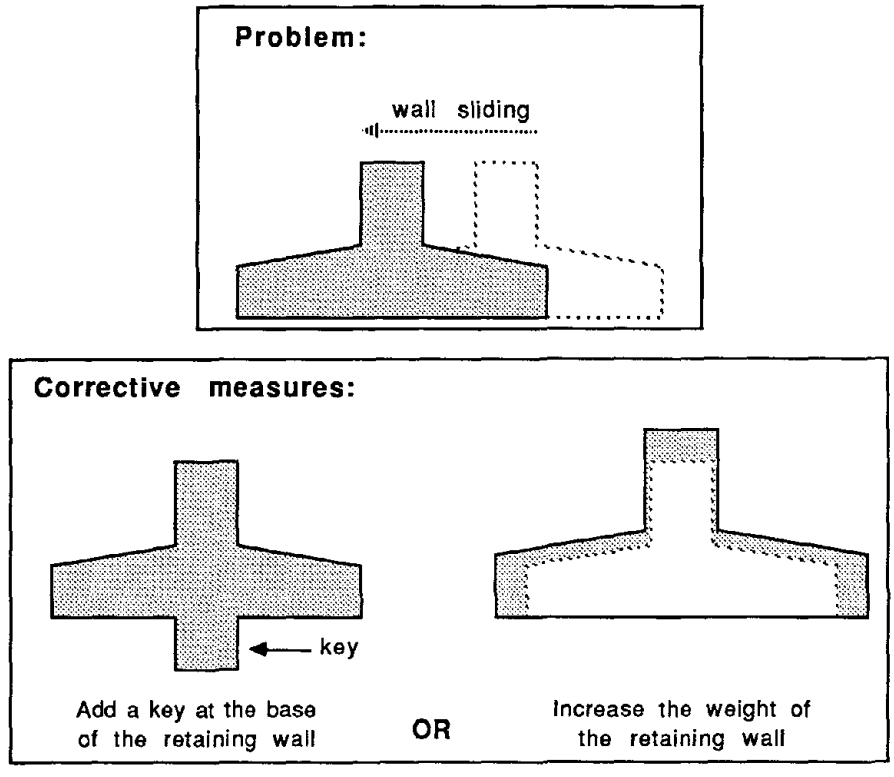


Figure 3-2: Measures for increasing the sliding stability of a retaining wall

and combinations of these alternatives to increase the resistance of the elements of the wall.

Some engineers argued that increasing the percentage of steel was not economical and that low cost was an important characteristic of their project, while others did not have the economic restriction and preferred not to increase the dimensions of the wall.

The simple engineering application presented in this section presents the following key characteristics:

- the system has to be used by multiple groups of users;
- a single application was going to be developed to be used by all the users;
- the users proposed the tasks to be performed by the system;
- all the users agreed with the procedures and methodologies to be taken in some of the tasks to be performed by the system; and
- for other tasks to be performed by the system, each individual user group had its own criteria and methodology and it was very difficult to achieve consensus between all the users to define a unified criterion to be embedded in a single application.

The kind of conflict described in this section is characteristic of many engineering situations. The

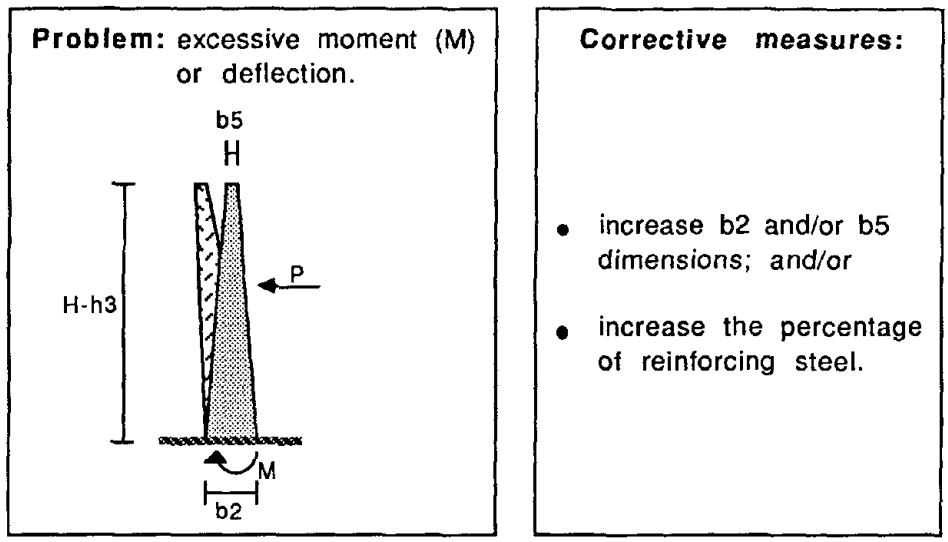


Figure 3-3: Excessive deflection or internal forces in the components of a wall

remainder of this section presents a framework for dealing with KBES applications with characteristics similar to the ones described in this section.

3.3 Proposed KBES Framework

This section presents a KBES architecture that addresses the key problems presented in the first section of this section, namely:

- the need for customization;
- the desire to avoid duplication of effort in developing a KBES in a particular domain by incorporating the common knowledge in that domain; and
- the ability to incorporate newly acquired, highly individual expertise.

The proposed framework for a customizable KBES is illustrated in Figure 3-4. The architecture shown in Figure 3-4 has the following key characteristics:

- it provides a repository of pre-compiled common knowledge in the domain;
- it allows a user organization to personalize the system by incorporating its own criteria and expertise; and
- new applications can be rapidly prototyped using the core knowledge base.

By splitting the knowledge base of a KBES into *common* and *custom* knowledge bases, the architecture provides a pool of commonly agreed domain knowledge and a separate repository for the personal expertise of individual users.

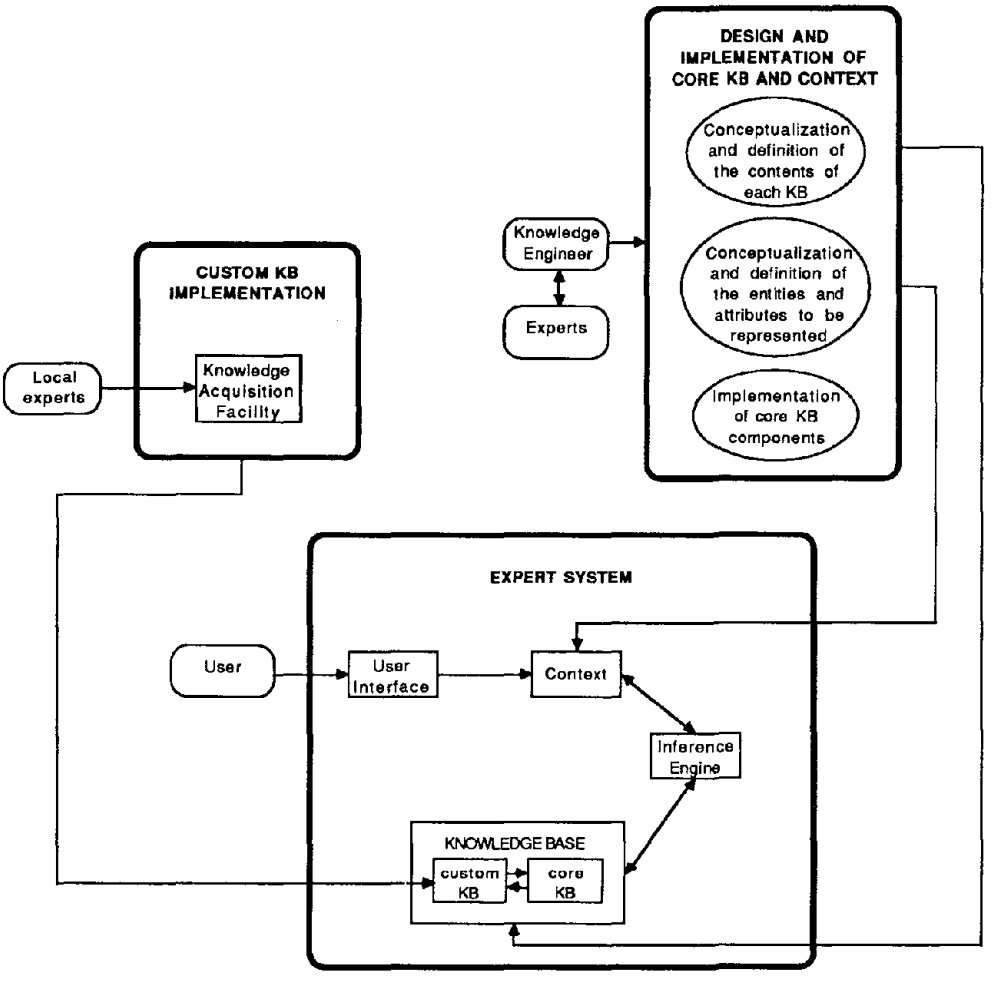


Figure 3-4: Overall architecture of the proposed framework

3.3.1 Components of Architecture

This section describes the components of the architecture shown in Figure 3-4.

Core Knowledge Base. The central component of the KBES is the core knowledge base. The core knowledge base contains the commonly agreed knowledge in the domain. This commonly agreed knowledge consists of at least a partial definition of all the different main tasks that can be executed by the KBES. The processes contained in the core knowledge base can be either heuristic or algorithmic. Any portion of the processes contained in the core knowledge base that is subject to individual interpretation must reside in the custom knowledge base.

Custom Knowledge Base. The custom knowledge base allows the individual users of the system to

personalize the processes that can be performed by the KBES. The custom knowledge base is used to allow each individual user to accommodate his/her own criteria in particular aspects of general processes where there is no consensus about the way these aspects must be interpreted or performed. The degree of personalization of the different tasks to be performed by the KBES is to be decided at the time of the definition of the contents of both the core and custom knowledge bases, and is a result of the consultation sessions with the different experts and/or end users.

An empty custom knowledge base will prevent the KBES from running since the contents of the custom knowledge base completes some of the processes contained in the core knowledge base. Therefore, a KBES developed with the approach suggested in this work can be distributed to the different individual users with *default* knowledge. This *default* knowledge may be modified by the user using the knowledge acquisition component of the KBES. In this way, the same KBES in the hands of two individual users with different opinions on the aspects that can be customized may produce substantially different results. By replacing the custom knowledge base of one individual user or organization by another one, the KBES may reproduce the individual expertise of individual users or organizations engaged in similar endeavors. The operation of replacing the custom knowledge base of an individual user by another one may be as simple as replacing a single file.

Knowledge Acquisition Facility. The knowledge acquisition facility is the component responsible for reviewing, modifying, and augmenting the custom knowledge base of the KBES. Through this component, individual user organizations can incorporate and update their specific expertise.

Context. The context is a model of the physical problem; it consists of a collection of facts that form an abstraction of the problem to be represented and all the factors involved in that problem. The context describes the problem to be solved as well as the current state of the solution process.

Inference Engine. The inference engine of a KBES applies the knowledge contained in the knowledge base to the context, which contains the description of the problem to be solved, and guides the application of these knowledge to arrive at a conclusion/solution. The inference engine component of a KBES should be independent of the contents of the knowledge base and must be built or selected by the knowledge engineer according to the nature of the problem to be solved by the KBES. Depending on the location of the application of interest in the derivation-formation spectrum of problems, a backward, forward or mixed backward-forward chaining strategy may be needed.

User Interface. The user interface is the link between the computer program and the outside world. A single program may have several interfaces that pass information to and from humans, other programs, data bases, display devices, or sensors.

3.3.2 Proposed Development Process

This section describes the four phases of the proposed development process.

Knowledge Base and Context Conceptualization. The development of a KBES following the approach shown in Figure 3-4 starts with the definition of the contents of the core and the custom knowledge bases by the knowledge engineer. The definition of the contents of each knowledge base (core and custom) is made, identifying the different processes that can be executed by the KBES and the aspects of the processes in which there is no general consensus of how they should be executed or interpreted. This identification of the contents of each knowledge base is made through consultation sessions with the different experts participating in the development of the KBES. In addition to identifying the contents of the core and custom knowledge bases, the knowledge engineer has to define in parallel the contents and structure of the context of the KBES. The definition of the context involves identifying the entities to be represented and their attributes, the relationships between these entities and an appropriate representation schema for such entities.

Once that the conceptualization of the contents of the two knowledge bases has been done (core and custom knowledge bases), appropriate knowledge representation schemes must be selected for both knowledge bases. The core knowledge base may contain tasks represented with different programming paradigms operating on different data structures. The representation scheme for the custom knowledge must be able to represent adequately the expertise and concepts of individual users and should keep this knowledge in a form that is easy to review and modify. The representation scheme for custom knowledge must also be adequate enough to allow the interaction of custom knowledge with the tasks contained in the core knowledge base.

The design of the knowledge acquisition facility of a KBES, the component responsible of modifying and reviewing the custom knowledge, depends on the representation scheme selected. A proper knowledge acquisition tool must present the custom knowledge in an easily understandable form and must allow the user to modify and incorporate this modifications to the custom knowledge base. The following section presents a representation scheme for custom knowledge.

Implementation of Core Components. Once the components of the core knowledge base have been identified and defined, they are implemented. Using the knowledge acquisition facility, a *default* custom knowledge base is created by consulting an expert or a set of experts. This initial custom knowledge base is used during the development of the KBES to allow the execution of the system.

Customization. Using the knowledge acquisition facility, the user groups modify the default custom knowledge base to include their own specific criteria and expertise.

Production. The production phase includes:

- the implementation of a proper inference engine;
- the implementation of an adequate user interface; and

- the integration of all the components of the KBES.

Once the system has been completed, tested, and distributed to a particular user, the custom knowledge base can be maintained using the knowledge acquisition facility.

3.3.3 Illustration

As an illustration, let us assume that the application described in section 3.2 had been implemented in the framework proposed. The following tasks, which are only a subset of all the tasks related to the application, would have been placed into the core knowledge base, since there was agreement between the experts:

- check the stability of the wall against sliding;
- modify the wall if the safety factor against sliding is not met;
- compute the deflection of a cantilever beam; and
- modify the wall if the maximum allowable displacement in the stem is exceeded.

The following concepts would have been placed into the custom knowledge base since a consensus was not reached among the experts:

- measures to be taken if the wall tends to slide;
- definition of the maximum allowable deflection in the stem of the wall; and
- measures to be taken if the maximum deflection of the stem of the wall is exceeded.

Figure 3-5 illustrates the tentative classification of some of the processes involved in the application described in section 3.2 and the relationship between the tasks placed in each knowledge base.

The engineers consulted for developing the application agreed in using the same method for computing the deflections of a cantilever beam (Newmark's method); however, there were disagreements in the measures to be taken in case that the maximum allowable deflection was violated. Similarly, the definition of the maximum allowable deflection may vary from project to project. Hence, the procedures that implement Newmark's method and the procedure that verifies that the maximum allowable deflection of the stem of the wall is not violated reside in the core knowledge base. However, the definition of the maximum allowable deflection and the measures to be taken if the maximum deflection is exceeded reside in the custom knowledge base so that the user can express his/her own criteria and modify it any time.

3.3.4 Control of Customization

In a large KBES, the access for modifying the custom knowledge base by the different users of the system may be restricted by a pre-defined access hierarchy of users. Thus, users with a low hierarchy may have the right to modify a limited portion of the custom knowledge base while users with a high access priority may access more and more relevant knowledge or criteria contained in the custom

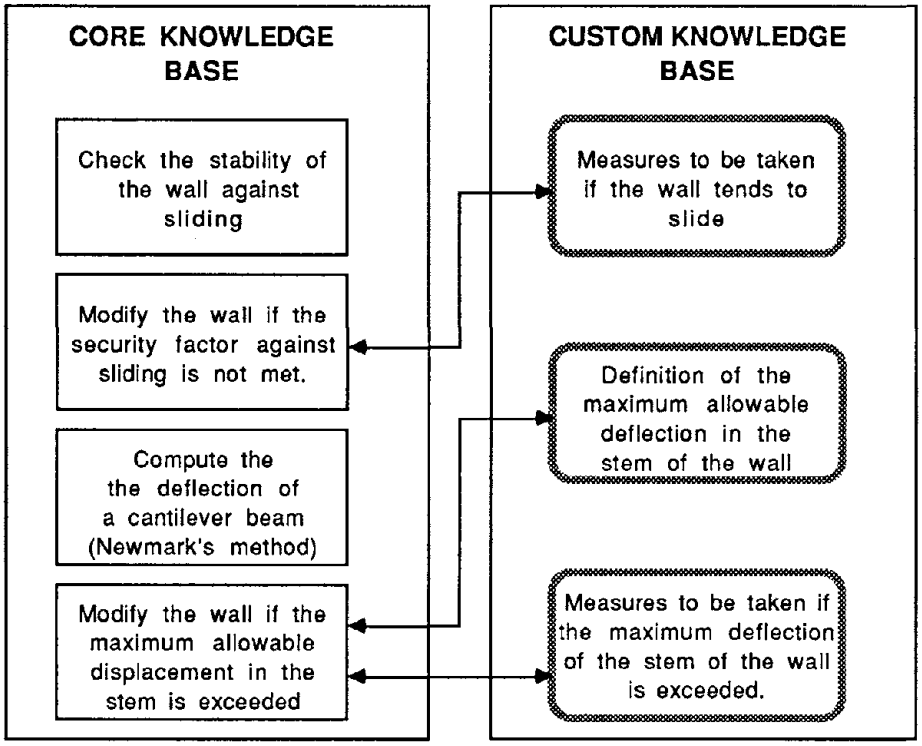


Figure 3-5: Tentative classification of some of the processes involved in the analysis and design of retaining walls

knowledge base. Similarly, the headquarters of a company may distribute the KBES to the different branches allowing them to modify only a portion of the custom knowledge base.

3.4 Conclusions about the Proposed Framework

The productivity of KBES developers can be greatly increased using core knowledge bases since experts providing the basic domain knowledge will be consulted only once. The work of building the core knowledge bases will not need to be repeated for subsequent applications, saving time of both the expert and the knowledge engineer. However, creating a core knowledge base for a host of applications is not an easy task since the core knowledge must be both general enough to be broadly applicable and specific enough to be valuable. Moreover, in some domains, it may be very difficult to draw a line between custom and commonly agreed knowledge.

The approach of consulting a number of experts for identifying the contents of the knowledge base of a KBES and defining what is commonly agreed knowledge and what is custom knowledge provides further advantages. The consultation sessions with the experts will eventually lead to a greater uniformity in the terminology and reduce the chances of misinterpreting domain concepts, leading to a better collective understanding of the domain. In the experiments performed by Boose [7] in which he combined multiple individual expertise for the creation of KBES, he demonstrated that the KBES with the best performance

were those in which multiple users worked simultaneously in the definition of the terminology and the attributes of the context to be represented in the KBES.

Although defining the contents of a core knowledge base is not trivial, once a solid base of several core processes is established, the productivity gained as new applications are produced should be significant. A similar approach has been successfully used in the domain of the retail brokerage business [19].

SECTION 4
KNOWLEDGE ACQUISITION FACILITY

This section presents the representation scheme selected for the custom knowledge of individual users and Eval-KAF, a knowledge acquisition facility developed for entering, reviewing and modifying the custom knowledge. Chapter 2 of [27] contains the details about the use of Eval-KAF.

4.1 Representation of Custom Knowledge

A key issue in the design of the custom knowledge base of a KBES is the selection of an adequate representation format for the custom knowledge of the individual users. The result of this selection provides fundamental guidelines in the design of the knowledge acquisition component of the KBES. The knowledge acquisition component must help users without a knowledge engineering background to capture, review and modify their own expertise. The knowledge acquisition component also needs a clear specification of the function of the custom knowledge required. An understanding of the ways the custom knowledge will be used provides useful guidelines for the design of an appropriate knowledge acquisition tool.

The schema selected for the representation of individual expertise is a *decision table* format. This selection was based on the following observations:

- individual expertise is reflected by:
  - the kinds of attributes, values, and conditions on these values entering into the left-hand (condition) side of rules; and
  - the alternate assignments of inferred attribute values or alternate paths of logical inference in the right-hand (action) side of rules;
- experts tend not to think about individual rules, but about tasks represented by clusters of rules containing various combinations of conditions leading to alternate solutions or inference for the task;
- the appropriateness and usefulness of decision tables was demonstrated in the work developed by Zozaya [47].

A decision table is an orderly presentation of the reasoning leading to a decision [13]. A decision table is conventionally considered as consisting of four quadrants, as shown in Figure 4-1. The upper left quadrant is the *condition stub* defining all logical conditions that have a bearing on the actions. The lower left quadrant is the *action stub* defining all possible actions. The upper right quadrant is the *condition entry* and the lower right quadrant is the *action entry.* The *rules* defining the logic of the decision table are the columns spanning the entry quadrants. A rule can be thought of as a logical AND function, that is, the rule is not satisfied unless each of the condition entries it contains is matched. The action entry indicates which actions are to be executed for each rule. In addition to the explicit rules in the table, a decision table can optionally have an *ELSE* rule. This *ELSE* rule will be executed if none of the rules in the

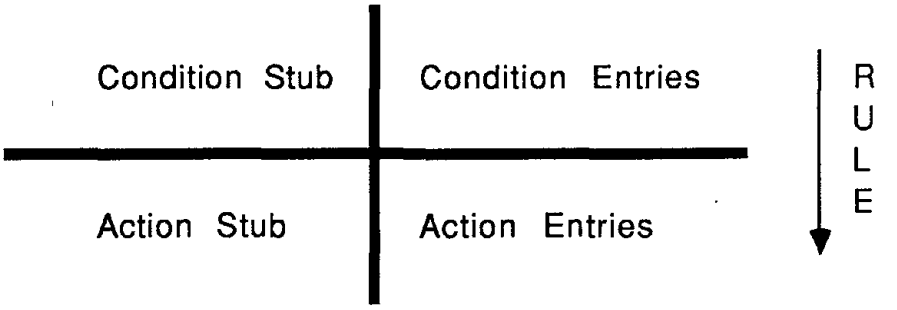


Figure 4-1: Conventional structure of a decision table

decision table are satisfied.

The most general form of a decision table, the *extended entry table*, can have multi-valued condition stubs. Another form, the *limited entry table*, only allows boolean condition stubs. The limited entry type of table lends itself to an examination of its logic. It is also easier to implement in computer programs than the extended form. Allowable condition entries and the character abbreviations representing the entries in limited entry decision tables are:

- *Immaterial* ("I" or "."); the rule is independent of the value of the condition in that row.
- *Explicitly True* ("T" or "Y"); the rule is satisfied only if the condition in the row is true.
- *Explicitly False* ("F" or "N"); the rule is satisfied only if the condition in the row is false.

In addition to the explicit entries defined above, a limited entry decision table can contain implicit entries, derived from non-independent conditions in a decision table. The implicit entries are:

- *Implicitly True* ("+"); the value of the condition in the row is predetermined as true by values of other conditions in the same rule.
- *Implicitly False* ("-"); the value of the condition in the row is predetermined as false by other conditions in the same rule.

According to Welland [45], the advantages of using decision tables over other representation techniques are that:

- the logic is stated explicitly and precisely in a compact form;
- a decision table can be easily modified to reflect a change of circumstances; and
- a decision table is easily comprehensible to a non-computer specialist.

One of the major applications of decision tables has been the representation of standards. A standard is a provision or normative statement stipulating that a product or process shall have or be assigned some quality [13]. In 1966, Fenves introduced the idea of using decision tables for representing the logic of a standard in a clear and systematic fashion [14]. Since then, research by a variety of organizations has been done to implement and continually improve upon this representational scheme.

4.2 Function of Eval-KAF

Eval-KAF is a tool constructed for building, reviewing and modifying a custom knowledge base consisting of a collection of individual knowledge sources represented as decision tables that can be linked together, forming a network of interdependent knowledge sources.

Eval-KAF was developed as the knowledge acquisition facility for EVAL, the application developed in this study and described in the following section. Eval-KAF can also be used as a general purpose knowledge acquisition tool in other applications in which the custom knowledge can be adequately represented in decision table format. Eval-KAF is implemented in ANSI C using a SUN3/60 workstation as the developing platform. Eval-KAF can perform the following operations:

- create new knowledge sources;
- erase existing knowledge sources;
- modify the information about existing knowledge sources;
- print the knowledge sources and the information associated with them;
- generate ANSI C source code that implements the knowledge sources; and
- generate a C program that can test individual knowledge sources.

The C source code generated by Eval-KAF as the implementation of the knowledge sources can be compiled and linked to other applications. This characteristic allows the knowledge sources to be called by any programming language that can call external C routines. Eval-KAF can help rule-based programmers to reduce the number of rules in a rule-based program and to introduce the possibility of modifying the domain knowledge of their application in a rapid and convenient way, without having to modify the source code of the program. The ability to modify a program rapidly can be important in some environments, and a program automatically generated from decision tables can be very flexible in this respect.

4.3 Use and Structure of Decision Tables in Eval-KAF

This section describes the anatomy of the different types of decision tables handled by Eval-KAF. This section also describes the process of generating a computer program from a decision table and the capability of associating multiple decision tables to represent complex concepts.

4.3.1 Types of Decision Tables

Decision tables are used to represent the individual knowledge sources that compose the custom knowledge base. Two types of knowledge sources[3] are used in Eval-KAF:

- *Assignment KS.* This type of knowledge source contains rules for assigning a value to a

[3]Using the work of Zozaya [47] as a precedent, from now on, the terms *knowledge source* and *decision table* are equivalent.

particular concept. The concept to be inferred is called the *main result* and the principal task of decision tables of the assignment type is to infer a value for the main result. The action stub of this type of knowledge source contains the possible values to be assigned to the main result. Associated with this type of knowledge sources there is control information, to be described later, that governs the way in which a knowledge source derives the main result.

- *General KS.* This type of knowledge source does not derive the value of a *main* concept. The action stub of this type of knowledge source does not contains a *main result,* but instead contains combinations of the different allowable actions that can be performed by the knowledge source. Control information is not applicable to this type of knowledge source; however, knowledge sources of this type can have different rule-firing strategies, as will be described later.

Figure 4-2 illustrates an *assignment knowledge source* that expresses the criterion for resizing a retaining wall in order to increase its weight. Figure 3-1 illustrates the dimensions of a retaining wall. The criterion contained in the knowledge source in Figure 4-2 can be summarized as follows:

NAME: increase_weight_of_wall	TYPE: assignment				
CONDITIONS	R1	R2	R3	R4	R5
vertical_height = "fixed"	T	F	T	F	F
base_size = "fixed"	F	T	T	F	F
height / base > 1.5	.	.	.	T	F
ACTIONS	R1	R2	R3	R4	R5
measure = "increase_base"	X			X	
measure = "increase_height"		X			X
measure = "increase_h1_h2_h3"			X		

Figure 4-2: Knowledge source of the *assignment* type

- **Rule 1.** If the height of the wall cannot be increased and the dimension of the base can, then increase it.
- **Rule 2.** If the dimension of the base cannot be modified and the height can be increased, then increase it.
- **Rule 3.** If neither the height nor the horizontal size of the base can be modified, then increase the vertical dimension of the base of the wall.
- **Rule 4.** If neither dimensions are fixed and the ratio *height/base* is greater than 1.5, then

increase the horizontal dimension of the base of the wall.

- **Rule 5.** If neither dimensions are fixed and the ratio *height/base* is less than or equal to 1.5, then increase the height.

The above criterion for determining the wall resizing strategy is expressed in the knowledge source as five explicit rules. Explicit rules are numbered from left to right and that is the order in which they are evaluated by Eval-KAF. It can be seen in Figure 4-2 that the action stub of knowledge source *increase_weight_of_wall* contains assignments to the same argument (*measure*). The argument *measure* is called the *main result* of the knowledge source. It can also be seen in Figure 4-2 that each rule has a single action entry; this is obvious since a variable can only contain a single value. Thus, only a single action per rule is allowed in assignment knowledge sources.

Figure 4-3 illustrates a *general knowledge source* that contains the knowledge needed to solve the quadratic equation:

$$AX^2 + BX + C = 0.$$

It can be seen from the figure that general knowledge sources do not contain a main result variable, instead, the action stub of a general knowledge source can have a variety of different actions. Furthermore, a rule in a general knowledge source can have multiple action entries. Rules in general knowledge sources are numbered from left to right and, as with the assignment knowledge sources, the rules are also evaluated from left to right. The actions in general knowledge sources are executed sequentially from top to bottom. Hence, an earlier action can compute an intermediate result that can be used by a later action in the same rule. This principle is applied in rules R1 and R3 of the knowledge source shown.

The knowledge sources are implemented by Eval-KAF as C functions. These functions can be called by any routine capable of interfacing with C functions. When a decision table is invoked, once that the table has been evaluated, the control **always** returns to the calling routine. The knowledge sources in Eval-KAF cannot alter directly the overall control of the application that makes use of the knowledge sources.

4.3.2 Evaluation of Rules

The assignment knowledge sources have a fixed rule-firing strategy, while the general knowledge sources can have different rule-firing strategies. Regardless of the rule-firing strategy, if none of the explicit rules are satisfied and the decision table contains an ELSE rule, the actions specified by the ELSE rule will be executed. If none of the explicit rules are satisfied and the decision table has no ELSE rule, Eval-KAF will exit the decision table printing a warning message.

In assignment decision tables, as soon as all the condition entries of a rule are satisfied, the action entries of that particular rule are executed and control is returned to the calling program. Thus, only a single rule can fire in assignment decision tables. In general knowledge sources, the user can select one of the following two rule-firing strategies:

NAME: solve_quad_eqn	TYPE: general							
CONDITIONS	R1	R2	R3	R4	R5	R6	R7	R8
A <> 0.0	T	T	T	T	F	F	F	F
B > 0.0	T	T	F	F	T	T	F	F
B*B-4*A*C >= 0.0	T	F	T	F	T	F	T	F
ACTIONS	R1	R2	R3	R4	R5	R6	R7	R8
D = B*B-4*A*C	X		X					
X1 = (-B-sqrt(D))/(2*A)	X							
X1 = (-B+sqrt(D))/(2*A)			X					
write("Complex roots")		X		X				
X2 = C/A/X1	X		X					
write("not quadratic")					X		X	
write("impossible")						X		X

Figure 4-3: Knowledge source of the *general* type

- *Fire one.* This strategy is identical to the one followed in the assignment knowledge sources: find the first executable rule, execute it and return.
- *Fire all.* This strategy starts to evaluate rules from left to right and it executes every rule that is satisfied.

The actions of a rule are executed starting from the top of the action stub and proceeding with the subsequent lower actions in the stub. In assignment knowledge sources, the main action is executed first and if there are additional actions in the rule, these are executed. As mentioned earlier, this characteristic can be used for using upper rows in the action stub to compute intermediate results used by actions defined in lower rows. Similarly, if the *fire-all* strategy is selected in a general knowledge source, earlier rules can be used to execute intermediate processes or to compute intermediate concepts that can be used by later rules in the knowledge source. Figure 4-4 illustrates a general knowledge source with a *fire all* rule-firing strategy.

The knowledge source in Figure 4-4 receives an argument that specifies whether a given material is concrete (*is_material_concrete*). The first two rules in the knowledge source prompt the user whether he/she knows the values of parameters *fc* and *fy* of the material in case that these parameters are

NAME: det_fc_and_fy	TYPE: general			
RULE FIRING STRATEGY: Fire all				
CONDITIONS	R1	R2	R3	R4
is_material_concrete = "yes"	T	T	T	T
is_fc_known = "yes"	F	.	T	.
is_fy_known = "yes"	.	F	.	T
ACTIONS	R1	R2	R3	R4
is_fc_known = ask the user	X			
is_fy_known = ask the user		X		
fc = ask the user			X	
fy = ask the user				X
fc_source = "provided by the user at run-time"			X	
fy_source = "provided by the user at run-time"				X

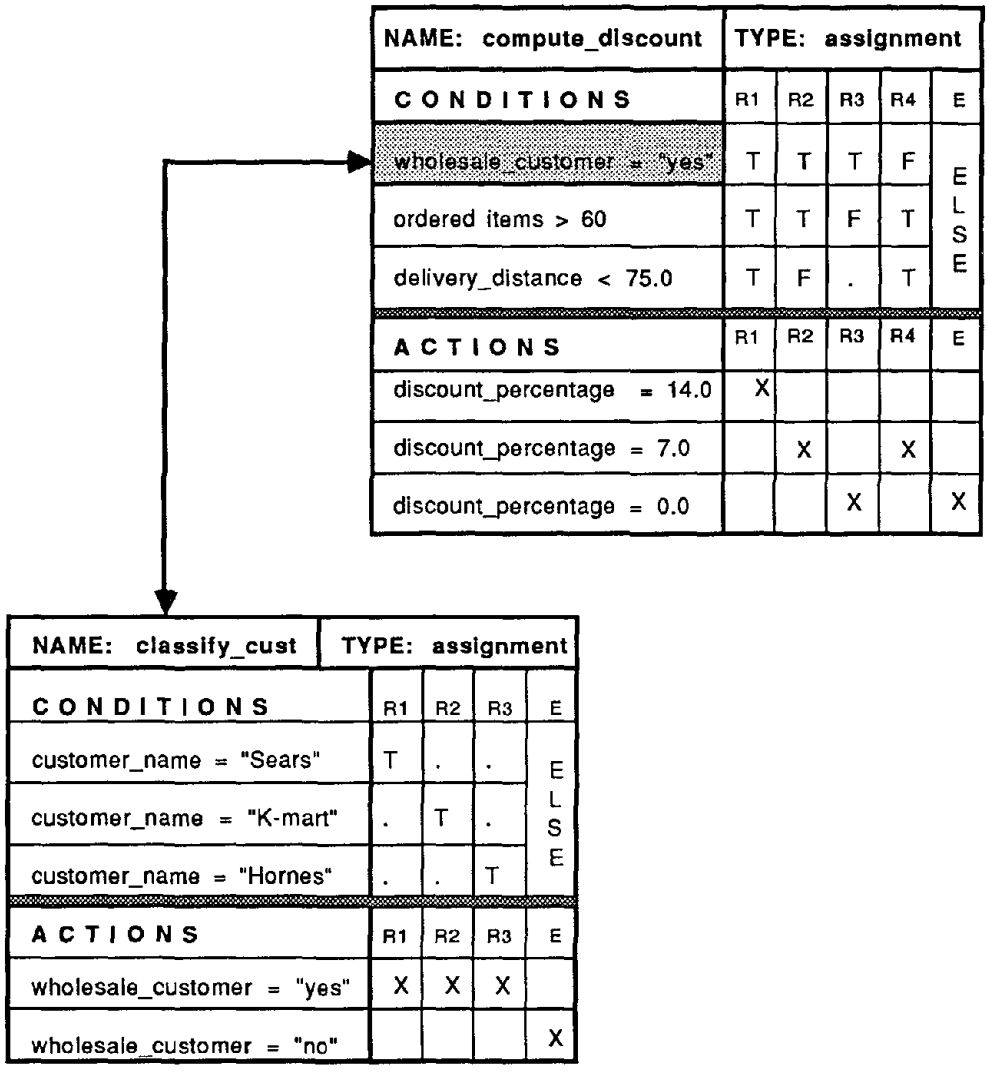
Figure 4-4: General knowledge source with a *fire all* rule firing strategy.

unknown. The results of those two prompts are stored in the arguments *is_fc_known* and *is_fy_known* respectively. The values of the arguments *is_fc_known* and *is_fy_known* are used in subsequent rules of the knowledge source. If the user replied that he/she knows the value of *fc*, the third rule performs two actions: prompt the user for the actual value of *fc*, and sets the value of the argument *fc_source* to *"provided by the user at run-time"*. The fourth rule is similar to the third one but works with the argument *is_fy_known*.

4.3.3 Evaluation of Networks of Decision Tables

Assignment decision tables can be used to derive the values of variables in other decision tables, as illustrated in Figure 4-5. The knowledge source *compute_discount* uses the value of parameter *wholesale_customer* in the condition stub for defining the criteria for assigning a discount percentage. The value of parameter *wholesale_customer* can be derived by another knowledge source, as illustrated in the figure.

If knowledge source *X* has parameters the values of which are derived by other knowledge sources, all the knowledge sources that are linked to parameters of *X* are executed before *X* can actually be evaluated. When *X* is invoked, the calling program passes the control of the program to *X* and *X* invokes each of the auxiliary knowledge sources that derive values of parameters of *X*. After an auxiliary



NAME: compute_discount	TYPE: assignment				
CONDITIONS	R1	R2	R3	R4	E
wholesale_customer = "yes"	T	T	T	F	ELSE
ordered items > 60	T	T	F	T	
delivery_distance < 75.0	T	F	.	T	
ACTIONS	R1	R2	R3	R4	E
discount_percentage = 14.0	X				
discount_percentage = 7.0		X		X	
discount_percentage = 0.0			X		X

NAME: classify_cust	TYPE: assignment			
CONDITIONS	R1	R2	R3	E
customer_name = "Sears"	T	.	.	ELSE
customer_name = "K-mart"	.	T	.	
customer_name = "Hornes"	.	.	T	
ACTIONS	R1	R2	R3	E
wholesale_customer = "yes"	X	X	X	
wholesale_customer = "no"				X

Figure 4-5: Association of knowledge sources.

knowledge source is executed, it returns control to *X*. After *X* is finally evaluated, *X* returns control to the calling program. Knowledge sources can be arbitrarily linked, the only restriction is that a parameter can only be linked to an assignment knowledge source.

The capability of networking decision tables is useful for breaking down a complex concept into different simpler concepts. The approach of decomposing a large decision table into smaller interrelated decision tables increases the transparency of the logic involved in deriving a concept.

4.3.4 Control Information

Assignment knowledge sources have control information attached to them. Control information governs the behavior of knowledge sources at run-time. Control information is divided into two parts:

- global control (affecting all the decision tables); and
- local control (affecting a particular decision table).

Each knowledge source contains two control flags that determine the way in which the knowledge source is executed. These control flags are part of the description of an assignment knowledge source and constitute the local control information. When an assignment knowledge source is invoked, it is by default evaluated in order to derive the result of interest. This is called the *automatic* mode of using a knowledge source. The automatic mode of execution can be changed with the first control flag: *volunteer*. The effects of the two possible values of *volunteer* on the behavior of the knowledge source are:

- *volunteer = OFF*: the knowledge source is automatically evaluated to produce the result of interest;
- *volunteer = ON*: the knowledge source stops as soon as it is invoked, presents to the user the value of all the condition variables and asks the user whether he/she wants to provide the value of the main result associated with the knowledge source; if the user agrees to provide the value, the knowledge source will exit and return the value provided by the user; if the user declines to provide a value for the main result, the knowledge source will be executed to provide the main result.

The effects of the two possible values of *override* on the behavior of the knowledge source are:

- *override = OFF*: after the knowledge source has been evaluated to derive the result of interest, control returns directly to the calling program;
- *override = ON*: after the knowledge source has been evaluated to derive the result of interest, the knowledge source consults the user about the validity of the result produced; if the user agrees with the result produced by the knowledge source, control returns to the calling program, otherwise, the user is prompted for the result to be returned to the calling program.

The local control information consists of the two previously defined control flags and cannot be modified at run-time. For each knowledge source, the values of each of these two flags is part of the description of the knowledge source. The user must use Eval-KAF to modify the control information of the knowledge sources.

The global control information is used to override the local control information of **all** the knowledge sources and is set at run-time. The application that makes use of the knowledge sources must either set or prompt the user for the values of two flags: *global_volunteer* and *global_override*. These flags are passed as an argument to all the knowledge sources and they carry what has been called the global

control information. The effect of the global control variables in the behavior of the knowledge sources is:

- *global_volunteer = OFF*: the knowledge source will behave according to the value of the local *volunteer* flag;
- *global_volunteer = ON*: the value of the local *volunteer* flag will be overridden regardless of its value and will be set to ON;
- *global_override = OFF*: the knowledge source will behave according to the value of the local *override* flag;
- *global_override = ON*: the value of the local *override* flag will be overridden regardless of its value and will be set to ON.

4.3.5 Explanation Capabilities

Assignment knowledge sources can use an explanation argument for storing a message that explains the reason that leads to the action taken by each rule of the knowledge source.

When the user creates a knowledge source of the assignment type, Eval-KAF asks the user whether he/she wants to add an explanation argument to the knowledge source. If the response of the user is affirmative, Eval-KAF adds automatically an extra variable to the knowledge source called *explanation* (*explanation* is a reserved variable name in assignment knowledge sources). The explanation variable can then be used in the additional actions stub to store a message with the justification for each of the actions of the rule. Figure 4-6 shows an assignment knowledge source with an explanation variable.

The *explanation* variable in assignment knowledge sources receives a special treatment in case that the control information of the knowledge source causes the main result variable of the knowledge source to have its value assigned by the user. The knowledge source will return one of the following values of the explanation variable:

- if the knowledge source is executed and the value of the main result variable is assigned by one of the rules of the knowledge source, the explanation variable will have the value that the user specified as an additional action in the corresponding rule; if the user does not explicitly assigns a value to the explanation variable in a given rule, the explanation variable will contain a NULL string;
- if the *VOLUNTEER* flag is on and the user agrees to provide the result of the main result variable prior to executing the knowledge source, the explanation variable will contain the message: *"volunteered by the user"*;
- if the *OVERRIDE* flag is on and the user overrides the value produced by the knowledge source, the explanation variable will contain the message: *"overridden by the user"*.

The explanation argument can be used by the application that calls the knowledge sources to keep track of the user interaction with the knowledge sources or to monitor the criteria contained in the knowledge sources.

NAME: Infer_fc_of_concrete	TYPE: assignment			
CONDITIONS	R1	R2	R3	E
year_built >= 1904 AND year_built < 1950	T	F	F	ELSE
year_built >= 1950 AND year_built < 1970	F	T	F	
year_built >= 1971	F	F	T	
ACTIONS	R1	R2	R3	E
fc_of_concrete = 3000.0	X			
fc_of_concrete = 3500.0		X		
fc_of_concrete = 4000.0			X	
fc_of_concrete = ask the user				X
explanation = "SEES -> a KBES written by J. F. Peters"	X	X	X	
explanation = "provided by the user at run-time"				X

Figure 4-6: Assignment knowledge source with an explanation argument

4.3.6 Conversion of Decision Tables into Executable Code

The knowledge sources, implemented as limited entry decision tables, are translated by Eval-KAF into C source code. This section describes how Eval-KAF converts a knowledge source into a C function and how this function evaluates the decision table.

There are two main approaches to convert a limited entry decision table into a computer program. The first approach consists of generating a decision tree from the decision table, usually by bifurcating the original decision table. The logic contained in a decision table can be expressed as a decision tree, which is a network with one condition at each node. The branches from each node represent the possible condition entries and the termination of each path, or limb, is a rule. Figure 4-7 gives an example of a decision tree generated from a hypothetical decision table.

The expression of logic in a decision tree strongly resembles a conventional flow chart. These *flow-chart-like* networks can then be compiled into a computer program. More than one decision tree can be generated from any decision table, depending on the order of selection of the conditions for testing, but it should be remarked that a unique set of condition entries will always isolate the same rule. Significant research has been made in the search for good tree-based algorithms (see [45] and [13]). Decision trees are a good tool for checking tables for uniqueness, completeness and, to a lesser degree,

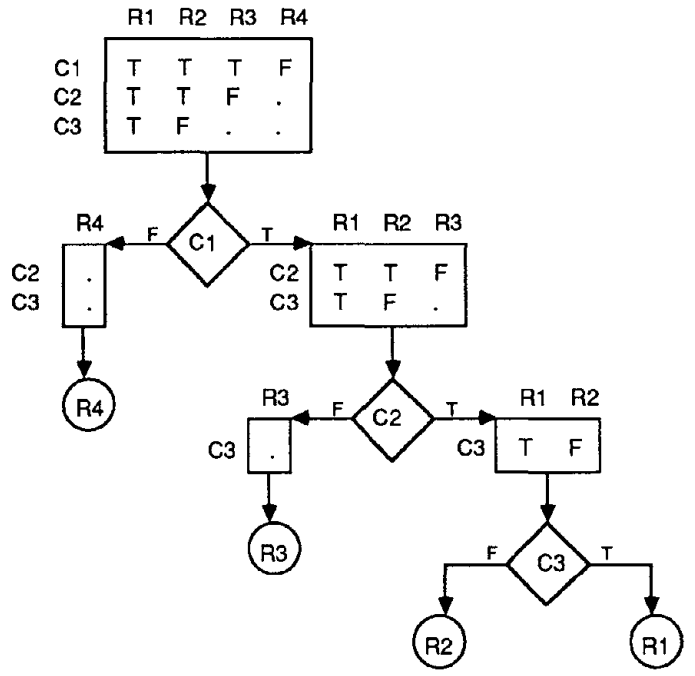


Figure 4-7: Generation of a decision tree from a decision table.

clarity.

In summary, the first major approach of converting a decision table into a computer program relies on decomposing the logic contained in a decision table into a decision tree. This decision tree is then used to compile the decision table into computer code.

The second approach is to encode the decision table in such a way that the conditions can be evaluated at execution time. This second approach differs from the first one as it relies on *interpreting* the decision table rather than compiling the table directly into code. The basis of the current methods for interpreting decision tables is the technique first advocated in a paper by Kirk [33], commonly called the *rule mask* method.

Kirk's technique will be described by giving an example of his algorithm applied to the decision table shown in Figure 4-8 [45]. From the original decision table shown in Figure 4-8, three binary matrices are generated, called the *mask* matrix, M, the *decision* matrix, D, and the *action* matrix, A. The mask matrix is coded so that there is a 1 for every position corresponding to a relevant entry in the table ("T" or "F") and a 0 otherwise. The decision matrix is created by inserting a 1 wherever there is a "T" entry in the corresponding decision table and 0 otherwise. Similarly, the action matrix is created by inserting a 1 wherever there is an "X" entry and 0 otherwise. These matrices are *static*, because they are not changed during program execution. For the example of Figure 4-8 the matrices shown in Figure 4-9a, 4-9b and 4-9c are generated.

C1	T	T	T	F
C2	T	F	F	.
C3	.	T	F	.
A1	X	X	X	
A2			X	
A3				X

Figure 4-8: Decision table used for illustrating the *rule mask* approach.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) (b) (c)

$$d = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

(d)

Figure 4-9: Mask matrices of the decision table in Figure 4-8.

At execution time, when the decision table is to be invoked, a *data vector*, d, has to be generated to reflect the current state of the conditions being tested. For example, if C1 is *true*, C2 is *false* and C3 is *true* at the time the decision table is to be executed, then d would be coded as shown in Figure 4-9d. This data vector, d, is then logically multiplied, or *ANDed*, with each of the columns of M in turn. The result of each of these operations is compared with the corresponding element of D (column) and if equality occurs, then the relevant rule is satisfied and the appropriate actions are executed. The actions to be executed are specified by the column of matrix A corresponding to the column of matrix D that matched the M^d vector. An example of this process is shown in Figure 4-10.

Depending on the rule-firing strategy being followed, this process of logical multiplication and comparison is continued until a rule is satisfied or until all columns in matrix D have been tested, meaning that all explicitly stated rules in the original table have been checked. Failure to find a match between any corresponding column of the logical combination "d ^ M" and D simply means that the ELSE rule applies.

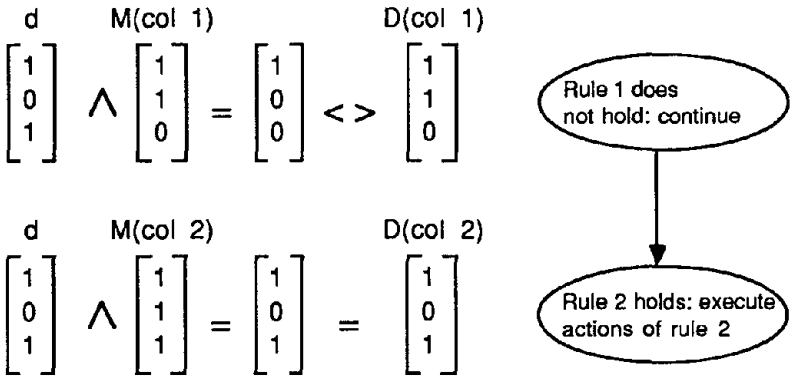


Figure 4-10: Logical multiplication and comparison of mask matrices.

Unlike most methods based on decision trees, Kirk's algorithm does not fail when the table contains an ambiguity. If two or more rules in the table overlap, the algorithm will simply assume that the first such rule encountered is satisfied. On the other hand, Kirk's algorithm makes no attempt to optimize the process of condition testing since all conditions have to be evaluated before applying the algorithm.

Eval-KAF uses the second approach, rule masking, for converting the knowledge sources into C functions. Eval-KAF generates two text files when converting the knowledge sources into C code:

- a file containing the mask (M), decision (D) and action (A) binary matrices of each knowledge source; and
- a file containing C source code with the necessary instructions to execute each of the knowledge sources; one C function per knowledge source is produced.

All the C functions that implement the knowledge sources have a similar structure. The execution process of a knowledge source is summarized in Figure 4-11. The necessary code for each of the individual processes executed by a knowledge source is generated by Eval-KAF.

4.4 Interaction Between Core and Custom Knowledge Bases

The custom knowledge base of EVAL consists of a collection of knowledge sources, implemented as decision tables, created and maintained by the knowledge acquisition facility developed for EVAL and described in the previous section.

The knowledge sources customize some of the tasks performed by EVAL. The different tasks of EVAL that can be customized and the degree of customization of these tasks will be described in the next section. This section describes the interface mechanism between the tasks in the core knowledge base and the knowledge sources that form the custom knowledge base.

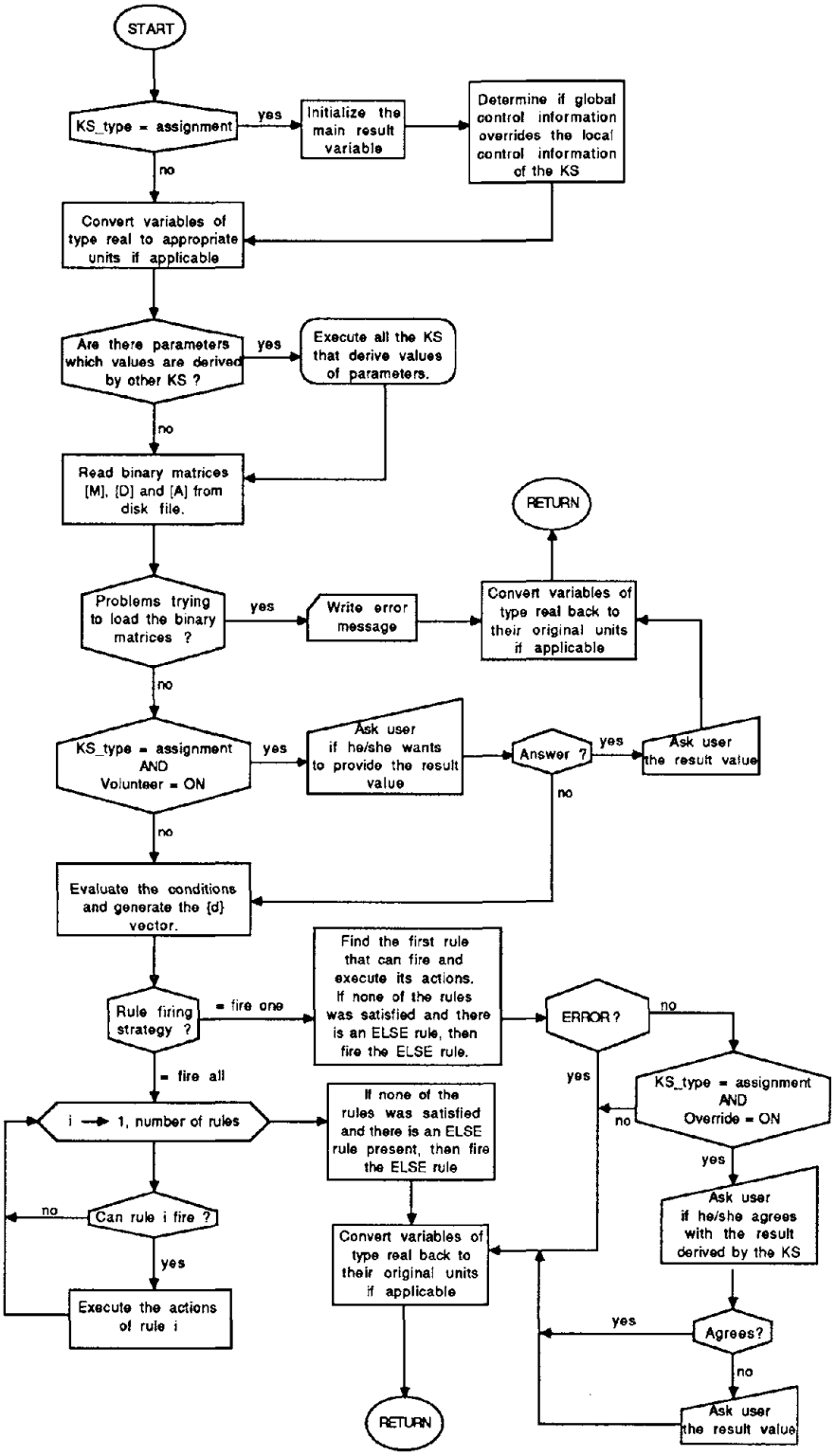


Figure 4-11: Summary of the execution process of a knowledge source

The knowledge sources work with variables, which carry the values needed for evaluating the knowledge source and the result or results produced by the knowledge source. When invoking a knowledge source, these variables have to be mapped to attributes of working memory elements (OPS83 objects) to retrieve and store the values needed and produced by the knowledge source.

Essentially, the mechanism for interfacing a core knowledge base task with a knowledge source is as follows:

- a rule is used to gather on the left hand side all the ingredients needed by the knowledge source;
- the right hand side of the rule calls the knowledge source, implemented as a C function, providing the knowledge source with all the necessary parameters it needs; and
- after the knowledge source has been executed, the results produced by the knowledge source are stored in their proper place in working memory; this correspondence is provided by the working memory elements bound on the left hand side of the rule.

In EVAL, the design of this interface is part of the construction of the core knowledge base. The design of such an interface consists of the following steps:

- once a portion of a given task is selected for customization, the parameters involved in the definition of the criteria for that specific sub-task are defined;
- an OPS83 rule is created for gathering on the left hand side of the rule all the attributes involved in the definition of the criteria for the given sub-task; these attributes are the necessary ingredients needed for constructing a rule or group of rules;
- a call to the C function that implements the knowledge source is prepared and the results produced by the knowledge source are stored in the appropriate working memory elements.

As an example, consider the problem of selecting the upgrading measures for reinforcing the stem of a retaining wall due to excessive deflection, illustrated in Figure 3-3. Selecting the measure for reducing the deflection in the stem of the wall is a task subject to individual judgment. Consider that, as a result of a survey among the different persons providing the expertise in the domain, the following attributes were identified as relevant for selecting the corrective measure to be taken:

- the dimensions of the beam that idealizes the stem of the wall: length and width of the two ends of the beam (a unit length of depth is assumed);
- the actual deflection of the wall and the maximum deflection allowed;
- the flexural moment acting on the wall and the percentage of steel in the cross section of the wall (per unit of depth); and
- the costs of reinforcing steel and concrete.

The above parameters are provided to the knowledge source as ingredients for defining the criteria for

selecting a corrective measure. These ingredients are used to build a knowledge source that reflects the expertise or preferences of a particular domain expert.

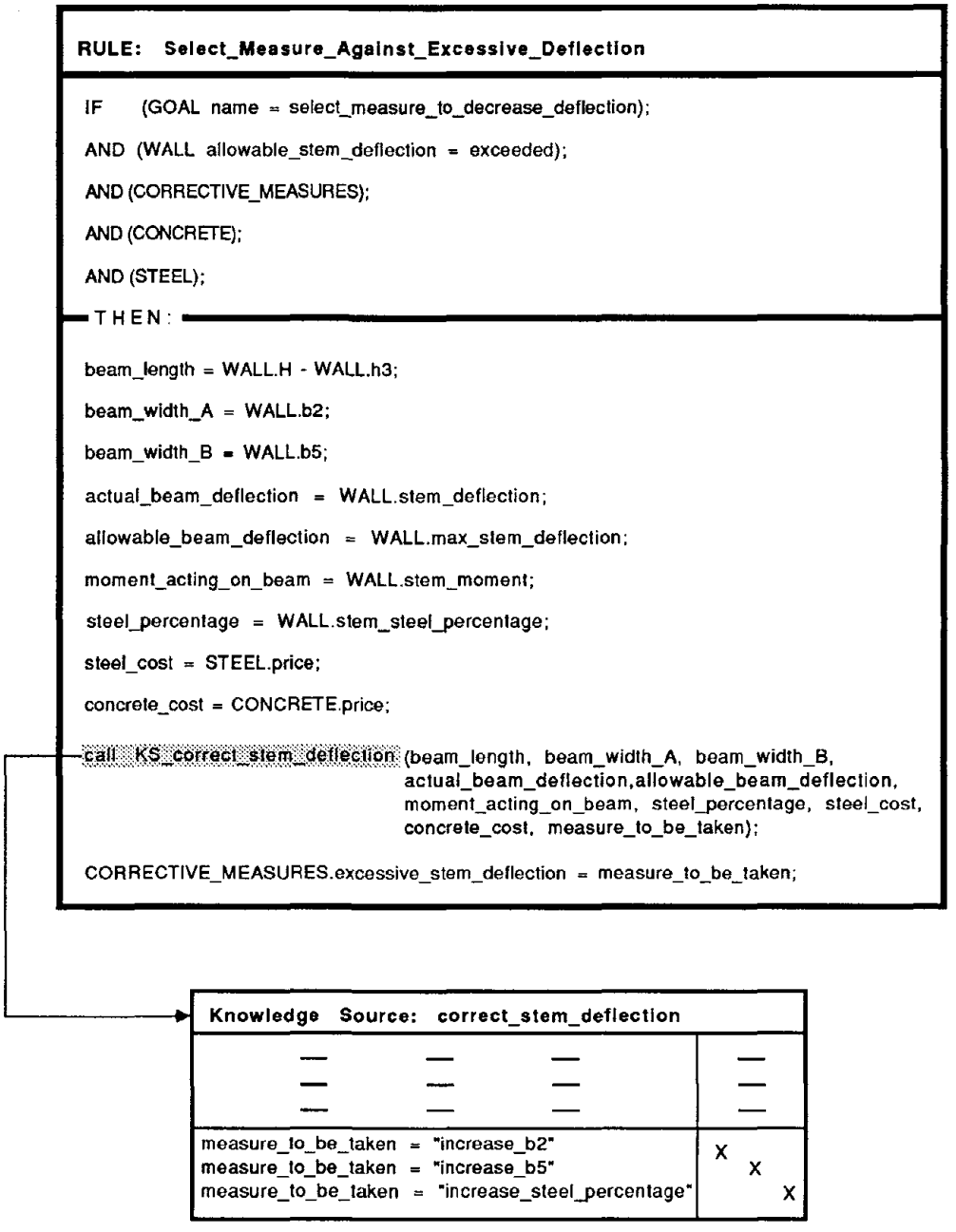


Figure 4-12: Interaction of a rule and a knowledge source

The rule shown in Figure 4-12 fires if the current active goal set by the control of the KBES is *select_measure_to_decrease_deflection* and the attribute *allowable_stem_deflection* of the wall has the

value *exceeded*. The remaining lines in the left hand side of the rule collect objects the attributes of which will be referenced in the right hand side of the rule.

In the right hand side of the rule in Figure 4-12, the ingredients needed to evaluate the knowledge source *correct_stem_deflection* are gathered from the objects bound in the left hand side of the rule. After the call to the knowledge source has been made, the results are mapped back to their corresponding attributes of objects. In the rule of Figure 4-12, only one result, *measure_to_be_taken*, is produced by the knowledge source and is stored in the attribute *excessive_stem_deflection* of the object *CORRECTIVE_MEASURES*. Any other result or possible results must be mapped back to working memory elements after the call to the knowledge source has been made.

Chapter 2 of [27] shows the interaction between an OPS83 rule and a C function implementing a knowledge source.

SECTION 5
EVAL: A KNOWLEDGE-BASED SYSTEM FOR EVALUATING THE SEISMIC RESISTANCE OF EXISTING BUILDINGS

This section presents a comprehensive description of EVAL, a KBES for evaluating the seismic resistance of existing buildings. EVAL is an expert system implemented according to the framework proposed in Section 3.

5.1 Introduction

The evaluation of existing buildings for seismic safety is a major technical and economic issue, particularly in the Eastern sections of the United States, where most existing buildings were designed and constructed without explicit consideration of seismic loading. The large building stock in this category makes it necessary to consider developing computer aids for addressing seismic safety on a broad scale. As states and municipalities introduce seismic design requirements for new buildings into their building codes and regulations, there will arise a corresponding concern for the seismic safety of existing buildings. Even if retrofitting on a large scale is not undertaken, the safety of critical buildings required to remain operational after an earthquake will become a major concern. Similarly, the safety of buildings undergoing rehabilitation and restoration, and thus falling within the scope of regulations for new buildings, will have to be addressed. In situations such as these, it will be important to ascertain the extent to which conservative design practices of the past may have provided a degree of reserve strength that can be counted on to resist seismic loads. On the other hand, factors such as age, corrosion, modifications, etc. may have contributed to reducing the original strength and integrity of these buildings.

Many aspects of evaluating existing buildings and recommending upgrading procedures are subject to experience and judgment. What constitutes the critical members and connections, what their condition and dependable seismic strength are, and what the most appropriate retrofit alternative is are all issues where assumptions and heuristics are required. Trying to develop a single algorithmic solution, with a *built-in* set of heuristics, would be inapplicable, or the resulting computer aid would be too cumbersome and restrictive. Thus, the use of KBES methodology is warranted. However, it would still be impossible to create a KBES with a single knowledge base that will produce evaluations and recommendations consistent with the experience and practice of all the different experts. Experts disagree on many aspects of the safety evaluation of a building and will want to use their own expertise in their evaluations.

EVAL is intended to address the issue of providing a general purpose computer aid for the evaluation of buildings, yet allowing each user organization to incorporate its own specific expertise. EVAL is implemented using the framework for customizable KBES described earlier in this work. The idea is to provide a common core containing general, commonly agreed knowledge and permit each user organization to expand and customize the knowledge base in order to accommodate its criteria and practices.

The need for customization in the domain of evaluation of the seismic resistance of existing buildings is illustrated in the following statements from the ATC-14 publication [2]:

- **7.1.5.2** The building has been provided with a redundant system for the transmission of lateral loads.
- **7.1.5.4.** There are no significant strength discontinuities in any of the vertical lateral force resisting elements.
- **7.1.6.4** The lateral force resisting elements form a well-distributed and balanced system that is not subject to significant torsion.
- **10.5.7** The exterior concrete or masonry walls are anchored to each of the diaphragm levels for out-of-plane loads.

Each of the underlined keywords is subject to individual interpretation. EVAL is intended to give each individual user organization the possibility of incorporating its own definitions of subjective concepts such as the ones presented above.

EVAL is implemented in two programming languages: OPS83 and C. OPS83 supports the rule-based programming paradigm and offers a comprehensive interface with routines written in C. Both OPS83 and C offer source code portability and they can be run on a number of different computers. One of the initial goals of this project was to be able to run EVAL on a PC platform in order to insure maximum portability of the system, allowing the distribution of EVAL among a diverse group of experts in the field of seismic evaluation of structures. The combination of OPS83 and C provided the desired portability, offering the means of developing the system on a workstation and then transferring the source code to a PC. However, the size of the system exceeded the capacity of PCs running under MS-DOS. This limitation forced the use of a workstation for developing and running EVAL, restricting its distribution.

5.2 Overview

This section presents the overall architecture of EVAL as well as the major tasks executed by the system.

5.2.1 Overall Architecture

The knowledge base of EVAL is divided into two components: the core, containing procedural and commonly agreed knowledge and a customized knowledge base, consisting of knowledge sources represented as decision tables, containing individual heuristics. The approach is to synthesize the commonly agreed knowledge into the core knowledge base and provide it as a primitive for the development of a more personalized KBES.

EVAL was implemented according to the framework illustrated in Figure 3-4 and described in Section 3. The knowledge acquisition facility and the representation of custom knowledge were described in Section 4.

The core knowledge base is the central component of EVAL and contains at least a partial description of all the tasks that can be executed by the system. Tasks are components of the evaluation process and are represented by procedures that execute these task. Some of the tasks in the core knowledge base are customizable via knowledge sources. The knowledge contained in the core knowledge base is represented in two paradigms: procedural and rule based.

Some of the tasks performed by EVAL are achieved by algorithmic procedures. Among these tasks are the organization and classification of the geometrical information of the building, as well as the reasoning about this information. The structural analysis capabilities of EVAL, as well as the algorithms for computing the strength of concrete sections also belong to this category.

Other tasks performed by EVAL are achieved using the rule-based paradigm. Tasks that involve pattern matching, such as the identification of nodes in the model of the building, or tasks that involve the evaluation of the compliance of several attributes with a set of conditions, are represented in the rule-based paradigm. The combination of rule-based and procedural paradigms considerably increases the problem-solving capabilities of an environment in comparison to a single-paradigm environment. Typically, an engineering problem has some major subproblems that are amenable to formal quantitative modeling and numeric computation techniques. However, some other subproblems, or the overall solution process, may require insight, qualitative reasoning, and/or pattern matching operations in order to obtain a solution or interpret the results of the solution process. Thus, it is only natural to expect that the coupling of symbolic and numeric computing in knowledge-based systems is needed to solve complex problems.

The custom knowledge base is a repository of knowledge sources that customize some of the tasks contained in the core knowledge base. Knowledge sources are represented as decision tables, as described in Section 4, and are implemented as C functions that evaluate these knowledge sources.

5.2.2 Major Tasks

This section presents a summary of the different evaluation processes that can be performed with EVAL. A detailed description of these processes is given later in this section.

EVAL can perform any of the following four seismic evaluation processes:

- quick evaluation of the building;
- evaluation of the transmission of seismic shears from the floor systems[4] to the elements that support these floor systems to determine the redundancy in the transfer of seismic loads;
- evaluation of torsion in floor systems; and
- evaluation of the resistance of individual elements at connections.

---

[4]Floor systems are also referred to as *horizontal subsystems*

At the beginning of a consultation session with EVAL, the user is presented with the following menu showing the above evaluation options.

```
Enter an option from the menu:

   1 -> Quick evaluation
   2 -> Evaluation of redundancy in horizontal subsystems
   3 -> Torsion evaluation of the building
   4 -> Reactions at the foundation and end forces of individual elements
   5 -> Check individual elements and connections
   6 -> Quit

Enter an option (1 - 6):
```

The user can select any number of evaluations from the above menu. Each of the evaluation tasks presented above has an associated list of subtasks. The entries on these lists may be repeated in more than one task. Once the user has selected the evaluation tasks to be executed, the control of the system determines the subtasks to be executed and the sequence in which they must be executed.

Each of the evaluation tasks displayed in the menu produces a written report with the results of the evaluation. The outcome of EVAL is a written report, contained in a text file called *eval-rep*, with the reports of all the evaluations selected by the user.

Except for the quick evaluation of the building (option 1 in the above menu), the remaining evaluation options require the execution of a structural analysis. If the user selects an option that requires structural analysis, he/she is given the option of releasing force components of selected members of the building prior to performing the evaluation(s) selected. Each member and each support joint is assumed to provide full fixity in all directions and all distortion components in a member are incorporated in the stiffness matrix of the member unless the user imposes a release on a force component of a particular member. This option may be used to reflect observations made during a site visit. If a particular connection is visibly deteriorated, the user may wish to release a force component of one or several of the members in that connection. EVAL uses an interactive dialogue with the user to select the structural members to be released.

5.3 Input

The task of evaluating the seismic resistance of an existing building starts by describing the building to be evaluated. The description of the building is performed by creating a text file containing commands in a Problem Oriented Language (POL) with arguments that describe the building and the seismic characteristics of the zone where the building is located. Detailed descriptions of EVAL's input POL and the process of creating an input file are presented in Chapter 3 of [27]. The initial description of a building is intended to be very simple, consisting basically of the following information:

- the name of the building;
- the definition of a three dimensional grid that provides the global geometry;
- the location of the individual elements (linear and planar);

- the available data on the geometric and structural properties of the different groups of cross sections of individual elements;
- the available data on the materials;
- the applicable seismic coefficients;
- the force and length units used in the preparation of the input file; and
- the location and magnitude of optionally applied external loads.

The above information will be described in the remainder of this section.

The geometry of the building is defined by a series of identical parallel horizontal (X,Y) grids as shown in Figure 5-1. In this representation, a grid point can be specified by giving the numbers of the X and Y axes that intersect at that point and the floor number that contains the point. This set of grids defines the boundaries of the building as well as the valid addressable points within the grid. Figure 5-1 illustrates a sample grid for defining the geometry of a building.

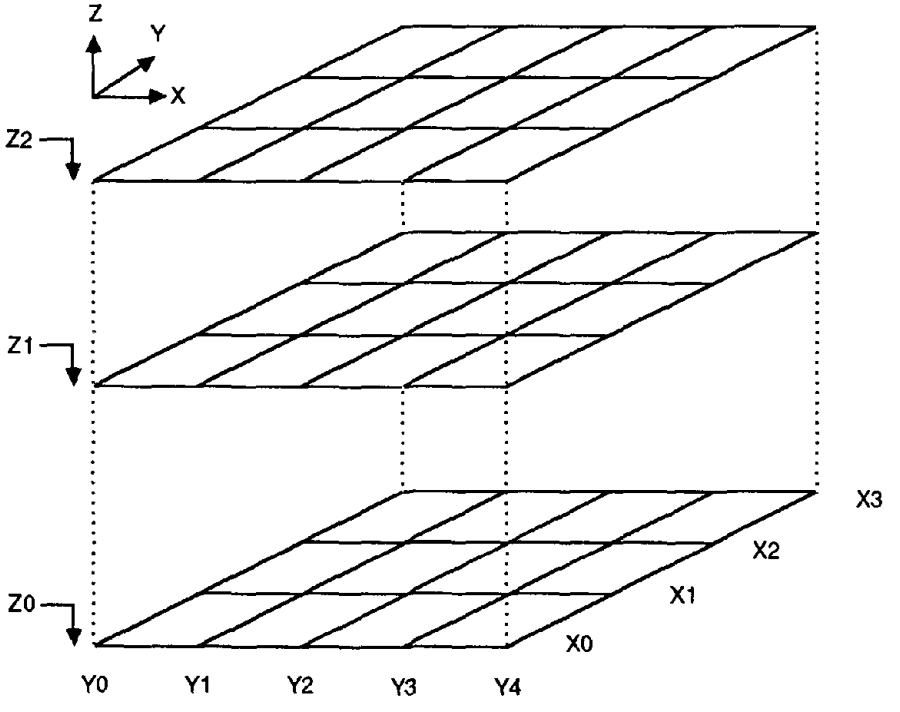


Figure 5-1: Sample grid for describing the geometry of a building

The grid shown in Figure 5-1 is used to specify the location of the individual elements. There are two types of individual elements: *linear* and *planar*. Linear elements are represented as straight lines and are defined by the end points of the line. Planar elements are represented as convex polygons and are defined by the number of vertices and their location within the grid. Planar elements are forced to be

convex polygons. If the user provides a concave polygon as input, EVAL will issue an error message and will abort the execution of the program. Convex polygons can be decomposed into several concave polygons as shown in Figure 5-2. This decomposition has to be done manually by the user when preparing the input file.

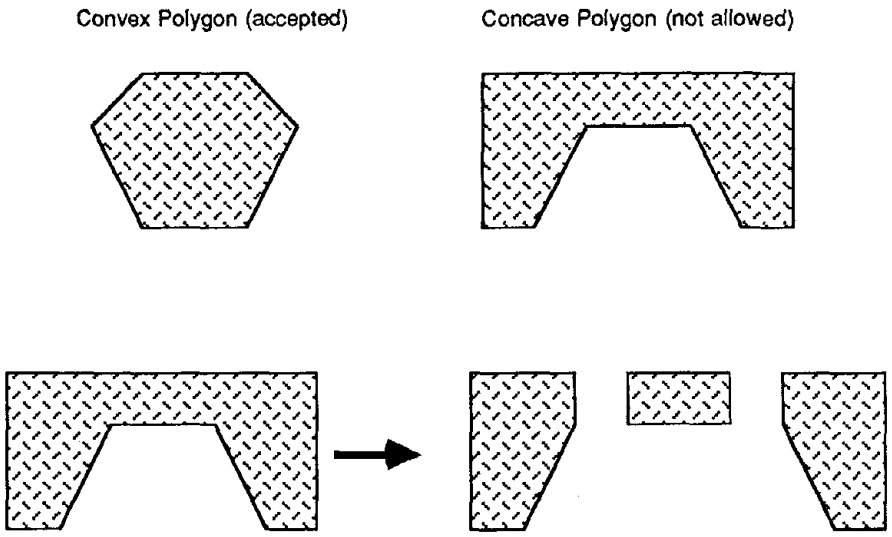
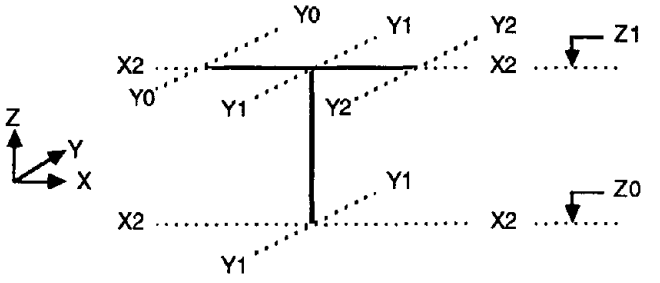


Figure 5-2: Convex and concave polygons

An individual element in a building is described not only by its location within the building; there is also a cross section and a material associated with the element. It is unlikely that the properties of the cross sections of the individual elements in a building will all be different or that the individual elements in a building will all be made of different materials. In order to avoid duplication, the information describing the elements is organized in a relational fashion. The number of different cross sections of individual elements must be identified by the user so as to provide the information on only these different cross sections. Similarly, the different materials that form the building must be identified and related to the different cross sections of individual elements. Figure 5-3 illustrates the way in which the description of the building are organized by the user.

The information about the materials and the cross sections of individual elements might not be complete, implying that some of this information will have to be derived or inferred by either EVAL or the user. The information that the user can provide about the materials present in the building is:

- the name or label of the material; it is required that the user provide a name or label in order to serve as a reference;
- the density of the material;
- Young's modulus of the material;



LINEAR ELEMENTS						
FROM			TO			Cross Section Ref.
X	Y	Z	X	Y	Z	
2	1	0	2	1	1	Col-Sec
2	0	1	2	2	1	Beam-Sec

LINEAR SECTIONS		
Label	Material	Dimensions and Properties
Col-Sec	Concrete	• • • •
Beam-Sec	Concrete	• • • •

MATERIALS	
Name	Properties
Concrete	• • • •

Figure 5-3: Relational organization of the description of the building

- the shear modulus of the material;
- the year the building was built; and
- the type of material (e.g., reinforced concrete, steel, etc.); depending on the type of material, parameters such as f'c, and fy may also be specified in the input file.

Except for the name of the material, the user may omit any of the above information relative to any of the materials in the building.

There are two types of cross sections: linear cross sections and planar cross sections. These two types of cross sections correspond to the linear and planar elements, respectively. The information that the user can provide about the linear cross sections is:

- the name or label of the linear cross section;
- the label of the material of which the cross section is made;
- the area of the cross section;
- the moments of inertia with respect with two orthogonal axes passing through the center of the cross section (Iy and Iz);

- the torsional constant (J) of the cross section;
- an optional angle of rotation (*alfa*) that rotates the cross section with respect to the principal axes of the element; this angle will be described later in this section; and
- the type of section (rectangular or circular); depending on the type of section, the width and height or the radius of the section will have to be provided.

The information needed for the planar cross sections is:

- the name or label of the planar cross section;
- the label of the material of which the cross section is made; and
- the thickness of the plates in this group.

In the case of the planar cross sections, all the information listed above is required.

The Iy and Iz parameters of linear cross sections are the moments of inertia of the section with respect the local Y and Z axes of the linear element. Figure 5-4 shows the local coordinate system of a linear element. The cross section of a linear element can be rotated with respect the principal axes as shown in Figure 5-5. The *alfa* angle is used to specify the orientation of the cross section with respect the principal axes of the element. If omitted, the value of the *alfa* angle is assumed to be zero.

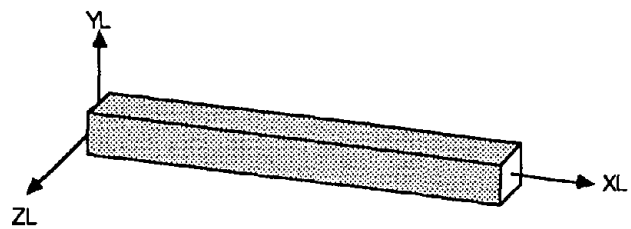


Figure 5-4: Local coordinate system of a linear element

There are two seismic coefficients required as input: *csx* and *csy*. Seismic forces acting on the building are idealized as statically applied lateral forces acting on each floor of the building. The coefficients *csx* and *csy* are used in the computation of the applied lateral forces. The coefficient *csx* is used to compute the lateral forces acting on the X direction (see Figure 5-1); similarly, the *csy* coefficient is used to compute the lateral forces acting on the Y direction. They are considered different since some earthquake design codes include not only the seismicity of the zone in these coefficients but also the characteristics of the structural systems in each of the two orthogonal directions [4].

As part of the input, the user must specify in the input file the force and length units used in the input data. All numeric quantities in the input file will be assumed to have appropriate units according to the length and force units specified in the input file. For example, if the user specified that the length units are pounds and the force units are feet, Young's modulus (E) of a material will have lb/ft^2 units; similarly, the

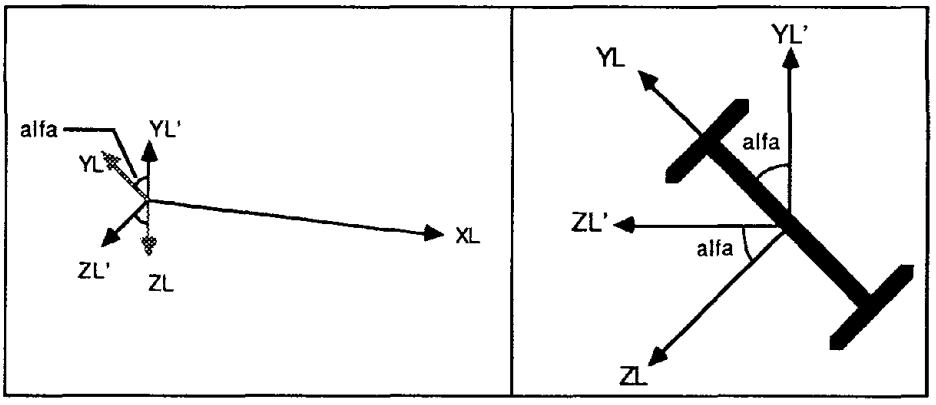


Figure 5-5: Rotation of a linear cross section with respect the principal axes

area of a cross section and the density of a material will have ft^2 and lb/ft^3 units respectively. If the user omits the specification of units in the input file, EVAL will assume the use of kilograms as force units and meters as length units.

Finally, the user can optionally specify in the input file gravity loads acting on the building. Gravity loads can be either concentrated or area loads and can represent appendages or mechanical equipment within the building. The convention for specifying these loads is described in Chapter 3 of [27].

5.4 Context of EVAL

This section describes the context of EVAL, consisting of a series of domain and control facts. Domain facts constitute the information that models the physical world. Control facts are the information used for managing the evaluation task.

5.4.1 Domain Facts

This section illustrates the overall representation of the building used by EVAL. Figure 5-6 shows the representation of the various types of information that describes the building and the relationships among these types.

The original input data provided by the user is processed by EVAL and new information is derived from it. As it will be described later, the geometric information in the original input is changed to a representation scheme of edges and vertices. EVAL contains a pre-processor that performs this change of representation and derives the information that is not explicitly provided in the input file, such as the connections between elements. The function of the pre-processor will be described in detail later in this section.

The information illustrated in Figure 5-6 constitutes the context on which the knowledge base of EVAL operates. Except for the information about edges and vertices, seismic coefficients, and units, the context

is represented in Object-Attribute-Value (O-A-V) triplets implemented as OPS83 objects. A detailed presentation of the objects that form the context is presented in Chapter 4 of [27].

The information about edges and vertices is stored in random access disk files. The number of edges and vertices can be very large and representing this information in the RAM will occupy large amounts of memory inefficiently. The routines for creating and accessing random access files are implemented in C. The seismic coefficients and the length and force units are scalar values and are stored in variables.

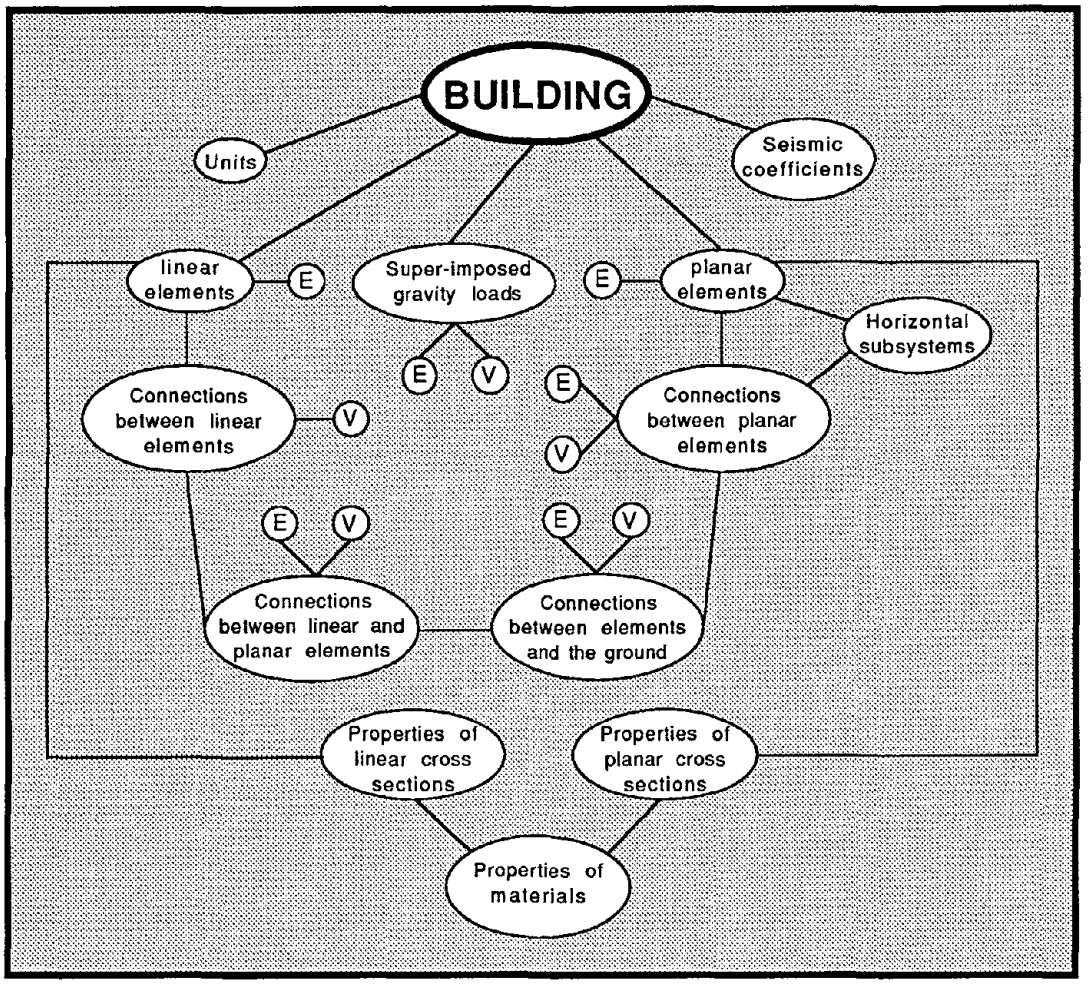


Figure 5-6: Domain context generated by the pre-processor

5.4.2 Control Facts

The control of EVAL is regulated with three classes of objects:

- *goal*;
- *goal_queue*; and

- *goal_sequence.*

The objects that implement the facts above are listed in Chapter 4 of [27]. The basic unit of control information is called a *goal.* A goal represents a task to be performed. A task consists of a sequence of related goals.

The tasks performed by EVAL are represented as lists of goals that have to be sequentially executed to complete each task. These lists of goals are represented as *goal_queues*, as shown in Figure 5-7. A goal queue is defined by its name, its status (*active* or *inactive*), and the active goal in the queue, that is, the goal currently being executed. A queue will have an active goal only if the status of the queue is *active*, otherwise none of the goals in the queue will be executed.

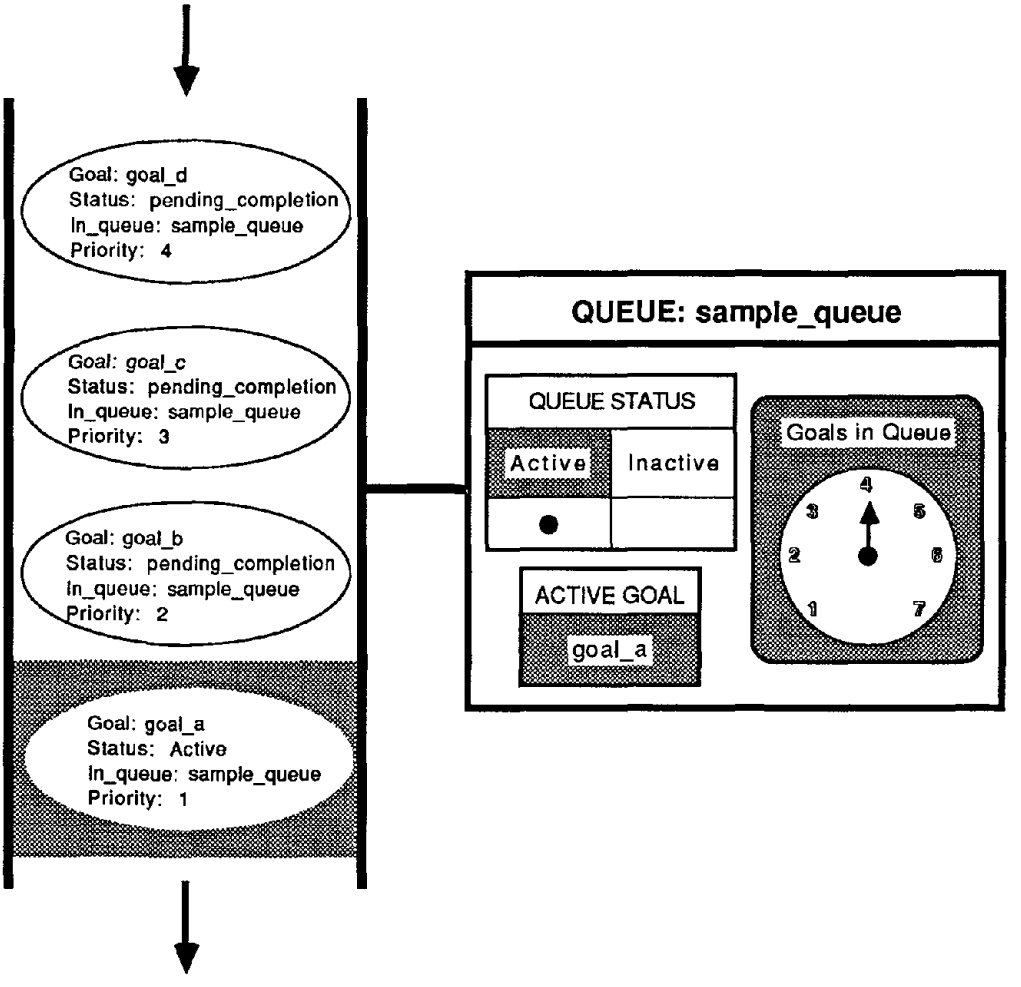


Figure 5-7: Queue of goals representing a task

Goals contain an attribute *in_queue* that references the queue to which they belong to. Additionally, goals have a priority assigned to them; this priority number is used to determine the position of the goal within the queue of goals. The goals in a queue are executed from lower to higher priority.

Since the user selects the evaluations to be performed at the beginning of a session, the goal queue representing all the necessary goals for performing the evaluations selected by the user is assembled at run-time. EVAL contains declarative information about the goals that need to be performed in order to execute each of the evaluation options. Besides determining the goals that need to be performed, EVAL has to determine the sequence in which these goals must be executed. The knowledge for determining the sequence of goals is stored in the objects *goal_sequence*.

5.5 Control

At the beginning of a consultation session with EVAL, the user selects from a menu the evaluation tasks to be executed. Each of the evaluation tasks displayed in the menu has a series of sub-tasks, represented as goals, associated with that particular evaluation task, representing all the activities that need to be performed. The information about the goals associated with each evaluation task, as well as the information about the sequence in which goals must be executed are static in EVAL.

After the user has selected the evaluation tasks he/she wants to perform, EVAL performs the following:

- create all the goals required to perform each evaluation task selected by the user;
- create the *goal_sequence* objects which contain the information about the sequence in which goals must be executed.

The goals that EVAL creates correspond exclusively to the evaluations selected by the user; the facts that determine the sequence in which goals must be executed correspond to all the different goals in EVAL. After creating the required goals and the information about the sequence in which goals must be executed, EVAL deletes and/or modifies the information about the sequence in which goals must be executed, depending on the goals created. To illustrate this operation, assume the following goal-sequence information:

- *goal_a* must be executed before *goal_b*; and
- *goal_b* must be executed before *goal_c*.

If, according to the evaluations requested by the user, *goal_b* is not required and is thus not created, the information about goal-sequences will be adjusted so that *goal_a* must be executed before *goal_c*, as illustrated in Figure 5-8. Similarly, the sequence facts in which the predecessor goal was not created are deleted.

Once that the goals to perform the evaluations selected by the user and the relevant goal-sequence information have been created, the sequence in which the present goals must be executed has to be determined.

The information about the goals to be executed and the sequence of execution form a directed graph. The edge direction reflects a precedence relationship between two nodes. Figure 5-9 illustrates a directed graph in which the nodes are goals and the edges are precedences in the execution of these goals. The directed graphs formed with the goal-precedence information in EVAL correspond to a *directed acyclic*

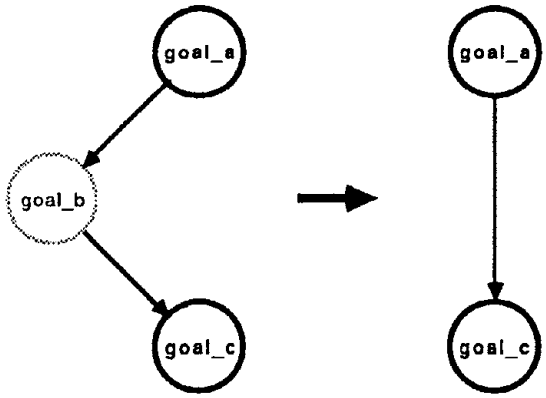


Figure 5-8: Modification of the information that determines the sequence in which goals must be executed

graph; the defining property of acyclic directed graphs [41] is that a cycle should never form when following the edges in the direction indicated. A cycle will represent an inconsistency in the definition of goal-precedences.

A fundamental operation on directed acyclic graphs is processing the vertices of the graph in such an order that no vertex is processed before any vertex that points to it. This operation is called *topological sorting*, because it involves ordering the vertices of the graph. In EVAL, the topological sort of the graph of goals is performed using the standard recursive *depth-first* procedure. Figure 5-9 illustrates the result of a topological sort of the graph in the figure. In general, the vertex order produced by a topological sort is not unique, since other legal topological sorts may be produced for the same graph.

The results of the topological sort operation is the sequence in which the required goals must be executed. These goals are placed in a goal queue, as illustrated in Figure 5-7. A consecutive priority number is assigned to these goals; goals with lower priority are executed first.

When the status of a goal in a goal queue is set to *active*, the goal is enabled for execution. EVAL uses a backward chaining strategy to execute goals:

- a goal is split into simpler subgoals;
- each subgoal is solved independently;
- the subgoals are then fused; and
- the original goal is solved.

When the status of the currently active goal in a goal queue is set to *completed*, the goal is eliminated from the goal queue and the goal with the next immediate higher priority attribute is made active. This operation is performed successively until the goal queue is empty.

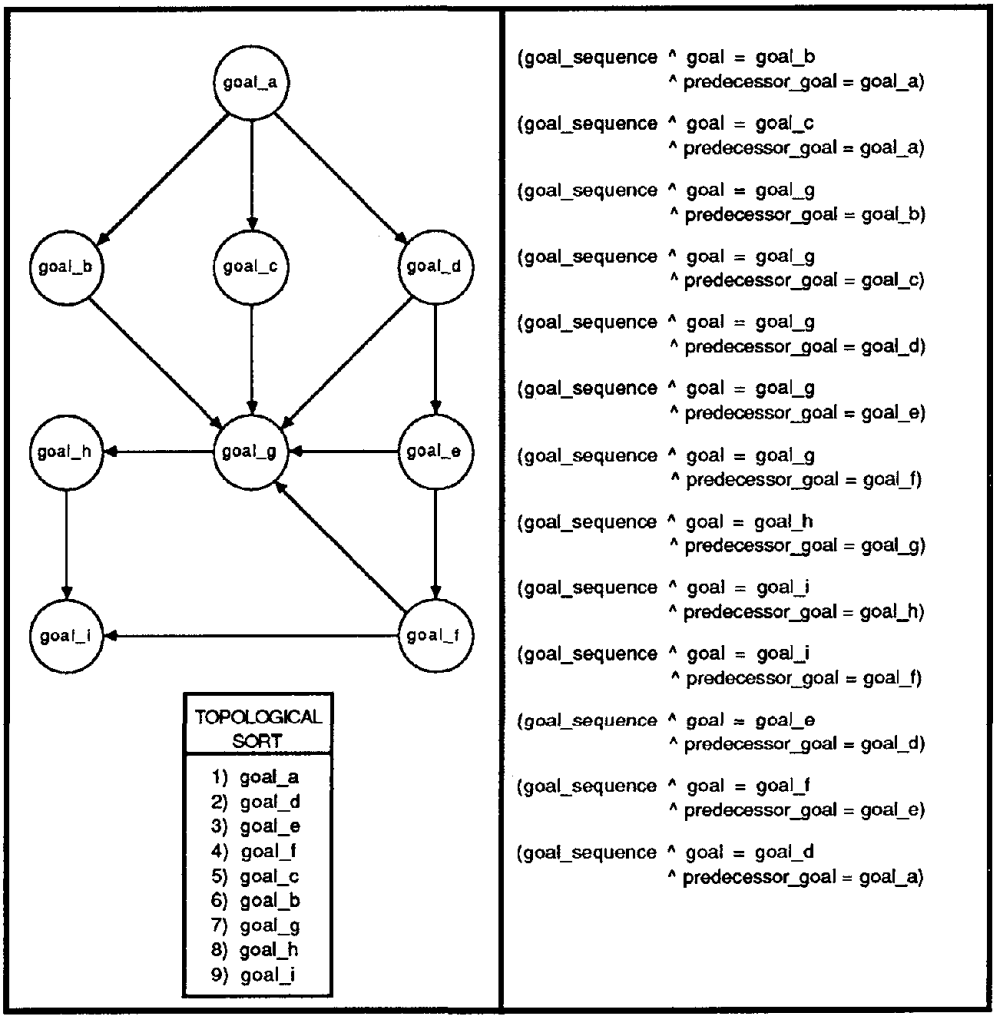


Figure 5-9: Topological sort of a graph formed by goals and precedences among these goals

5.6 Core Knowledge Base

This section describes the tasks contained in the core knowledge base of EVAL. Some of these tasks can be customized via knowledge sources. The customization of these core tasks will be detailed in the next section.

5.6.1 Generation of the Domain Context

The first task of EVAL is a pre-processor that reads the input file created by the user, checks its correctness and consistency, and generates the context used by the knowledge bases of EVAL. This section describes the sub-tasks performed by the pre-processor and the building representation created

in the context.

Classification of Individual Elements. The individual elements are classified according to their orientation.

Linear elements are classified into one of the types listed in Figure 5-10. The end points of the line that defines the linear element are reordered according to the criteria specified in Figure 5-10. Figure 5-11 illustrates the different linear elements that can appear in a building.

Linear element defined by its end points: (Xa,Ya,Za) and (Xb,Yb,Zb)		
Type	Conditions	End points swapped if
column	Xa = Xb Ya = Yb Za <> Zb	Zb < Za
beam_xx	Xa = Xb Za = Zb Ya <> Yb	Yb < Ya
beam_yy	Ya = Yb Za = Zb Xa <> Xb	Xb < Xa
beam_xy	Za = Zb Xa <> Xb Ya <> Yb	Xb < Xa
diag_xz	Xa = Xb Ya <> Yb Za <> Zb	Yb < Ya
diag_yz	Xa <> Xb Ya = Yb Za <> Zb	Xb < Xa
diag_xyz	Xa <> Xb Ya <> Yb Za <> Zb	-

Figure 5-10: Classification of the linear elements in a building

Planar elements are classified into one of the types listed in Figure 5-12. Figure 5-13 illustrates the different types of planar elements. The equation of each individual plate is computed using three vertices of the plate. The equation of a plate is used by EVAL to identify the location and orientation of the plate in space and to find the connections between the plate and other elements. EVAL makes sure that the three vertices picked to compute the equation of a plate are not colinear, otherwise the equation of the plate cannot be computed. The general equation of a plate is:

$$AX + BY + CZ + D = 0.0,$$

and the coefficients A, B, C and D are computed according to the following formulas:

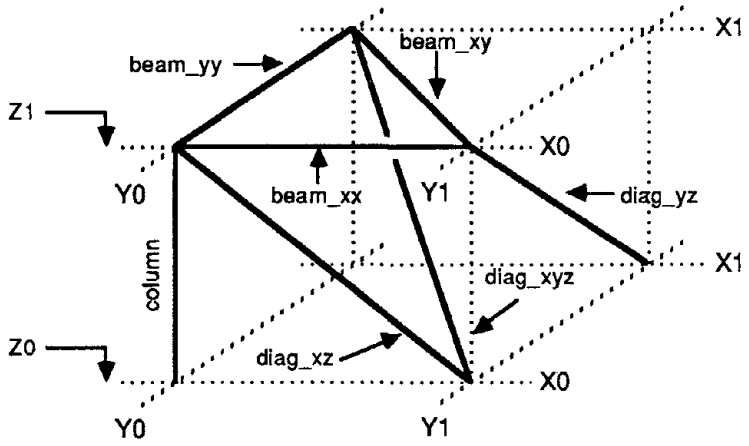


Figure 5-11: Illustration of the different types of linear elements

Point 1: (X1, Y1, Z1)
Point 2: (X2, Y2, Z2)
Point 3: (X3, Y3, Z3)

Ap = (Y2-Y1)*(Z3-Z1) - (Y3-Y1)*(Z2-Z1)
Bp = (Z2-Z1)*(X3-X1) - (Z3-Z1)*(X2-X1)
Cp = (X2-X1)*(Y3-Y1) - (X3-X1)*(Y2-Y1)
Dp = -(Ap*X1 + Bp*Y1 + Cp*Z1)

IF(Dp < 0.0) sign = 1 ELSE sign = -1

$$S = \sqrt{Ap^2 + Bp^2 + Cp^2} * sign$$

$$A = \frac{Ap}{S} \qquad B = \frac{Bp}{S} \qquad C = \frac{Cp}{S} \qquad D = \frac{Dp}{S}$$

Geometric Representation Scheme. The user originally describes linear elements by specifying their end points and planar elements by specifying their vertices. EVAL changes the original representation of the elements, creating geometric data tables that contain the geometric properties of the elements, organized to facilitate processing of individual elements. Geometric data tables contain boundary coordinates and parameters to identify the spatial orientation of the plates (polygon surfaces). Geometric data can be organized in several ways [24]. A convenient method for storing coordinate information is to create three lists: a vertex table, an edge table, and a polygon table. Coordinate values for each vertex in the object are stored in the vertex table. The edge table lists the endpoint vertices defining each edge. Each plate or polygon is defined in the polygon table as a list of component edges. This scheme is illustrated in Figure 5-14. The surface table contains pointers back to the edge table, which, in turn, contains pointers back to the coordinate values in the vertex table.

Listing the geometric data in three tables, as in Figure 5-14, provides a convenient reference to the components (vertices, edges and polygons) of an object. The representation shown in Figure 5-14

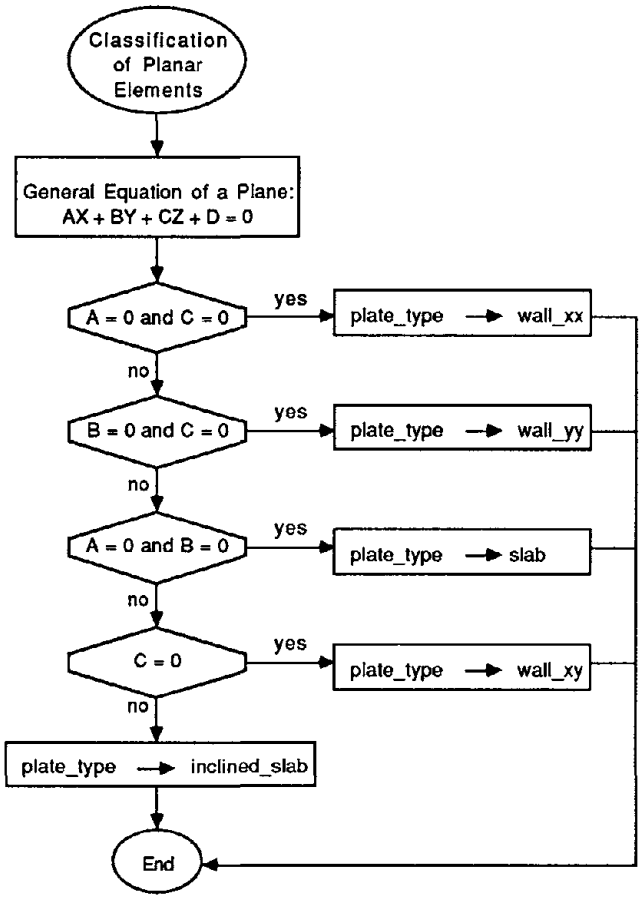


Figure 5-12: Classification of planar elements in a building

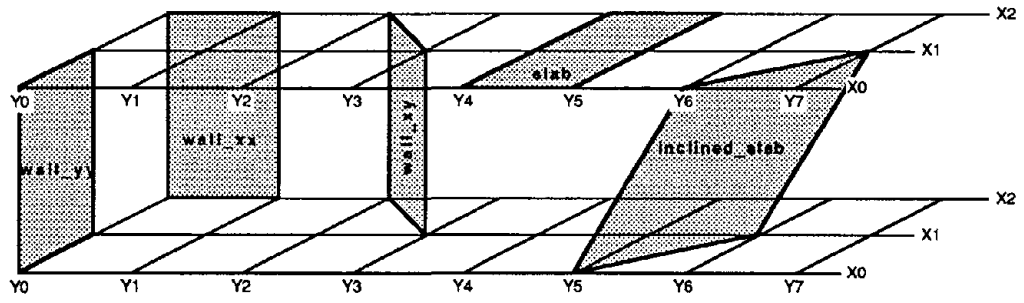


Figure 5-13: Illustration of the different types of planar elements

prevents the duplication of information for vertices that appear in more than one edge and for edges that appear in more than one surface.

Initial Identification of Vertices and Edges. Using the original description of the building provided by the user, EVAL changes the internal representation of linear elements, planar elements and externally

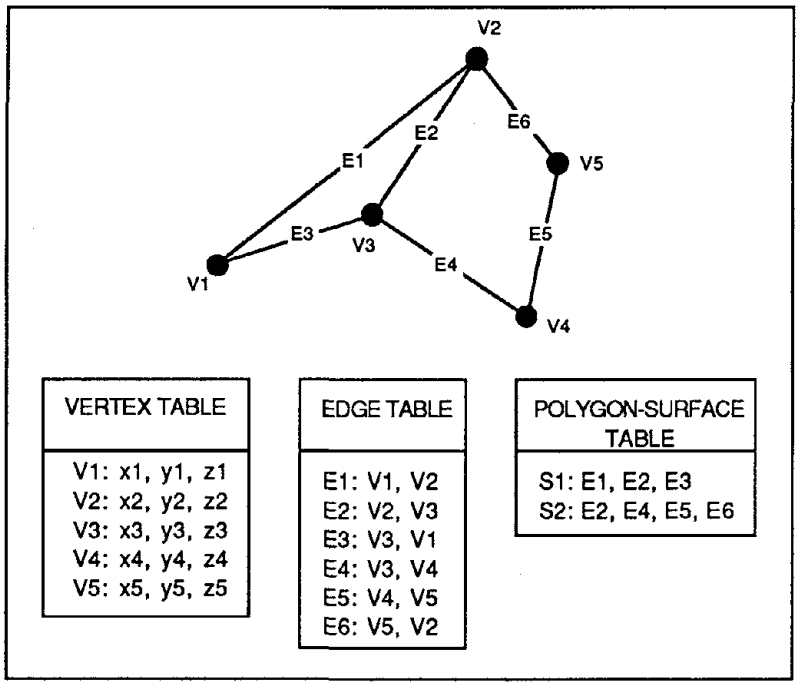


Figure 5-14: Geometric data tables for representing individual elements

applied loads, into the scheme shown in Figure 5-14. EVAL recognizes the following as a vertex:

- a grid point which is the endpoint of at least one linear element;
- a vertex of a planar element;
- a vertex of a surface load; and
- a grid point where a concentrated load is applied.

Similarly, the edges are:

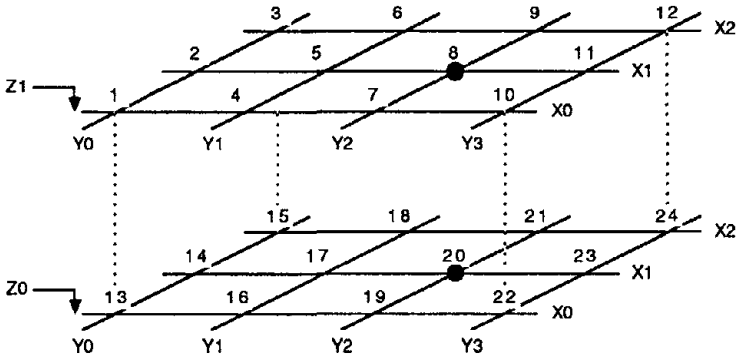
- linear elements;
- sides of polygons defining a plate; and
- sides of polygons defining a surface load.

The above lists of vertices and edges are updated during the process of finding the connections between individual elements, since connections are also represented as vertices or edges. Not all grid points are vertices. To keep track of the grid points that correspond to vertices and the enumeration of these vertices, an internal table is created that identifies whether a grid point corresponds to a vertex or not.

The user specifies the address of a grid point by providing three integer numbers: the X and Y axes that intersect in the point and the floor number or Z level where the point is located. The maximum

number of grid points in the three-dimensional grid is known, and if a convention for numbering the grid points is established, the (X,Y,Z) address of a grid point can be compressed into a single integer. This number, representing the address of a grid point, can be easily expanded into the original (X,Y,Z) address of the grid point by re-tracing the convention used.

Figure 5-15a illustrates a three dimensional grid and the convention followed by EVAL to number the points inside the grid. Figure 5-15b illustrates the conversion operations between the two different representations for a grid point. Figures 5-15c and 5-15d illustrate the tables used to keep track of the grid points that correspond to vertices.



(a)

grid_num(X1,Y2,Z1) = 8	Xaxis(8) = 1 Yaxis(8) = 2 Zlev(8) = 1
grid_num(X2,Y1,Z0) = 18	Xaxis(18) = 2 Yaxis(18) = 1 Zlev(18) = 0

(b)

NODE_TO_VERTEX		
Node	is_vertex	vertex_ID
1	false	0
2	false	0
.	.....	.
8	true	1
.	.....	.
20	true	2
.	.....	.
24	false	0

(c)

VERTEX_TO_NODE	
vertex_ID	node
1	8
2	20

(d)

Figure 5-15: Different representations for grid points. Tables for mapping grid nodes and vertices

Identification of Connections Between Individual Elements. All connections between individual elements are identified by the pre-processor. This identification is done in four steps:

- connections between linear elements;
- connections between planar elements;

- connections between linear and planar elements; and
- connections between linear and planar elements and the ground.

Connections between linear elements are represented as vertices and are located by finding the intersection between two linear segments. Connections between plates can be represented by either a vertex or an edge, as shown in Figure 5-16a. It is assumed that two coplanar plates do not overlap; defining two coplanar plates that overlap is an error detected in the stage of reading the input file. Connections between linear and planar elements, as well as connections between these elements and the ground, can also be represented by either a vertex or an edge. Figure 5-16b shows several types of connections between linear and planar elements.

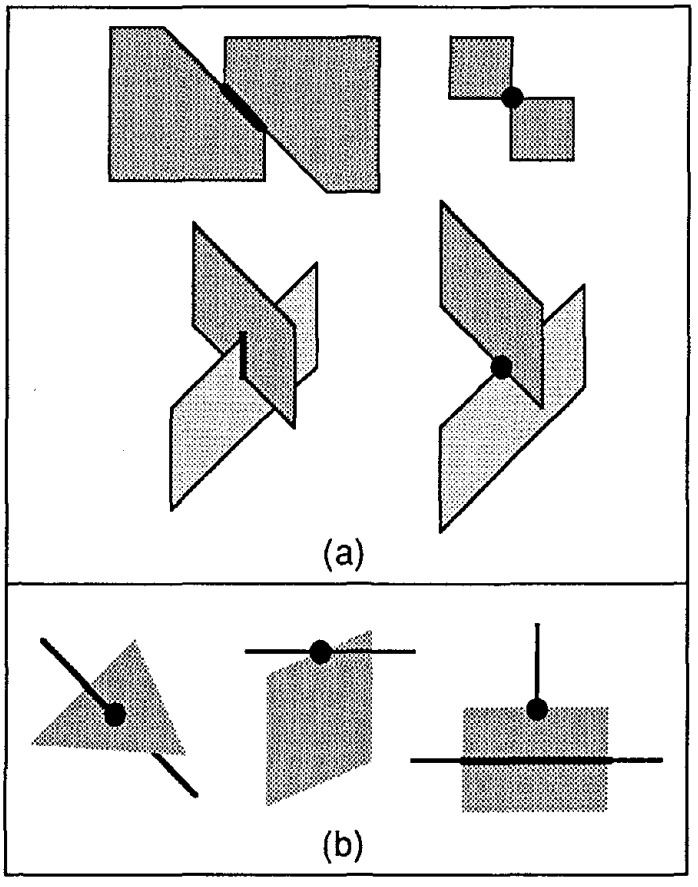


Figure 5-16: Intersections between individual elements

Generation of Local Edges. Once all the connections have been identified and the tables of vertices and edges have been completed, a second table of edges is generated. The reason for generating this second table is that edges representing linear elements must not contain more than two vertices. When the user specifies the addresses of individual elements in the building, he/she is not aware of any

intermediate vertices between the end points of a linear element or on a side of a planar element. After identifying the connections between elements, an original edge that defines a linear element can contain several intermediate vertices between the endpoints. It is necessary that a linear element contain only two endpoints; otherwise, when performing structural analysis, the contribution of that element to the stiffness will be improperly considered in the mathematical model.

Figure 5-17 illustrates the generation of local edges. The table of edges shown in Figure 5-17b corresponds to the three linear elements in Figure 5-17a. Edge E1 corresponds to the *column* placed by the user between the points (X1,Y0,Z3) and (X1,Y0,Z0); these two points were labeled by EVAL as V1 (vertex 1) and V6; similarly, E2 corresponds to the *beam_yy* between V2 and V3 and E3 corresponds to the *beam_xx* between V4 and V5. The connection between the column and the beam_yy created a new vertex (V7), located at (X1,Y0,Z2). The column represented by E1 has two intermediate vertices: V7 and V4. In structural analysis, it is necessary to consider the column not as a single element spanning from V1 to V6, but as three contiguous column-segments, spanning respectively from V1 to V7, V7 to V4, and V4 to V6. These segments of the original edge are called *local edges* and the original edge is called *global edge*. The table shown in Figure 5-17c shows the table of local edges for the elements shown in Figure 5-17a; an auxiliary table, shown in Figure 5-17d is created to map the local edges of a global edge.

Identification of Horizontal Subsystems . A plate with a concave shape can be defined as a series of convex polygons, as shown in Figure 5-2. The last process performed by the pre-processor is to group all the slabs connected to each other into single units called *horizontal subsystems*. The algorithm for grouping all the elements topologically connected is presented in Chapter 4 of [27].

Another reason for defining separate contiguous slabs is the case when there are adjacent slabs of different materials or adjacent slabs with different thicknesses. A horizontal subsystem is formed by either a single slab not in contact with any other slab, or by a group of slabs topologically connected. Horizontal subsystems are revised later in the evaluation task to determine whether they can behave as a single rigid unit. This process will be described later in this section.

The information generated about the horizontal subsystems includes:

- the number of slabs in the horizontal subsystem and their identification;
- the identification of the connections between the slabs that form the horizontal subsystem;
- the identification of all the connections between the slabs in the subsystem and other elements; this information is used for determining the redundancy in the transmission of the seismic forces acting on the horizontal subsystem.

5.6.2 Quick Evaluation of the Building

This task is the simplest evaluation that can be performed on a building. The quick evaluation procedure is not subject to customization and consists of the computation of the following:

- the total weight of the building;

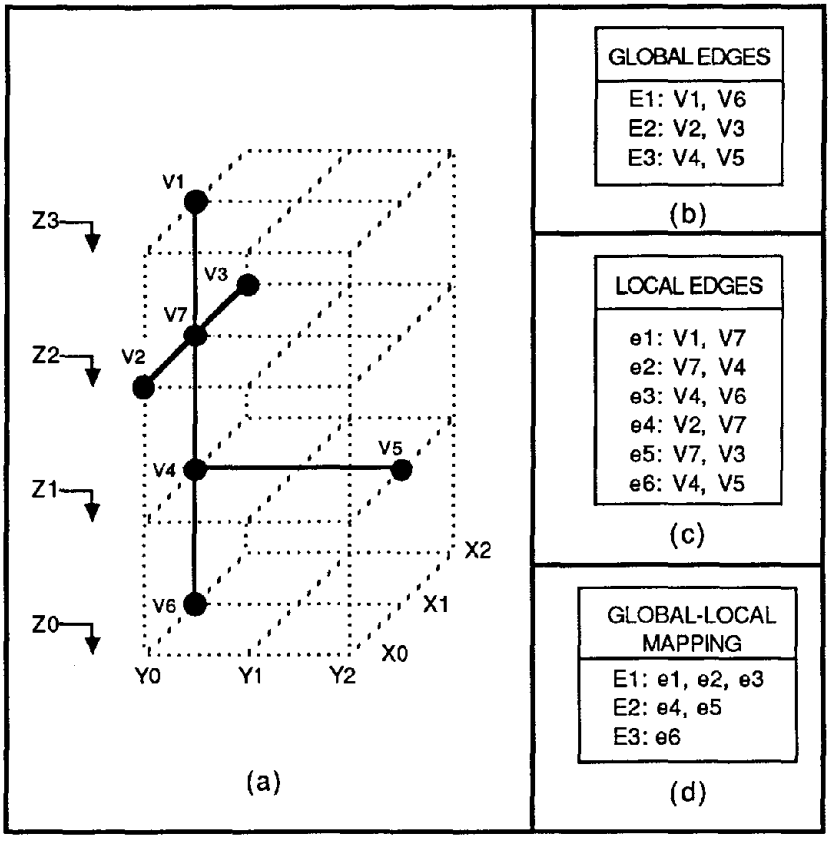


Figure 5-17: Global and local edges

- the base shears acting on the building along the X and Y directions; the procedure for computing the seismic loads acting on the building is described later in this section;
- the overturning moments in the X and Y direction; and
- the center of gravity of the building.

5.6.3 Determination of Material and Cross Section Properties

The user might have incomplete information about properties of materials and cross sections of linear elements. The properties of sections of planar elements are required to be provided as input data. The properties of materials needed are:

- Young's modulus (E); and
- shear modulus (G).

The properties of cross sections needed are:

- area (A);

- moments of inertia (Iy and Iz); and
- torsional constant of the section (J).

The above properties of materials and cross sections are the ingredients needed for computing the stiffness matrices of individual elements. Additionally, if the material is reinforced concrete, the following properties need to be known in order to compute the resistance of elements made of concrete:

- *f'c* of concrete and *fy* of reinforcing steel;
- percentage of reinforcing steel in the concrete; and
- depth of the concrete cover of reinforcing steel.

If any above property is missing for any given material, cross section, or concrete material, it has to be inferred in order to continue with the evaluation task. EVAL keeps track of the sources that provide or derive the values of material and cross section parameters.

The process of determining the needed properties of materials and cross sections is summarized in Figure 5-18. The goal is to derive the values of:

- Young's (E) and shear (G) moduli of a materials;
- geometric properties of cross sections required for the computation of the stiffness matrix; and
- the properties and parameters that define a reinforced concrete material.

If any of these parameters is not provided as input, it has to be inferred. In the current implementation, if a material is reinforced concrete, Young's and shear moduli can be derived from the properties of concrete (f'c). Several knowledge sources control the criteria for inferring the parameters mentioned above. Inferences for other types of material can be incorporated if there is sufficient knowledge for inferring E and G. In the case of properties of cross sections, if any of the necessary geometric parameters are not provided as input, EVAL has to either compute them or request them from the user. If there is not sufficient information for computing a missing parameter, the value of this parameter will have to be provided by the user at run-time.

5.6.4 Evaluation of Horizontal Subsystems

The pre-processor of EVAL groups interconnected slabs into units called horizontal subsystems (HS). Horizontal subsystems are considered rigid (infinitely stiff) units formed by one or more slabs; this assumption is used in the idealization for structural analysis. The horizontal subsystems identified by the pre-processor need to be evaluated to insure that they behave as rigid units. Horizontal subsystems not considered rigid are either decomposed into smaller groups of slabs or classified as *flexible* subsystems, with no in-plane stiffness.

The process of evaluation of a horizontal subsystem is summarized in Figure 5-19. The sub-tasks of the evaluation task that are subject to customization are highlighted in the figure.

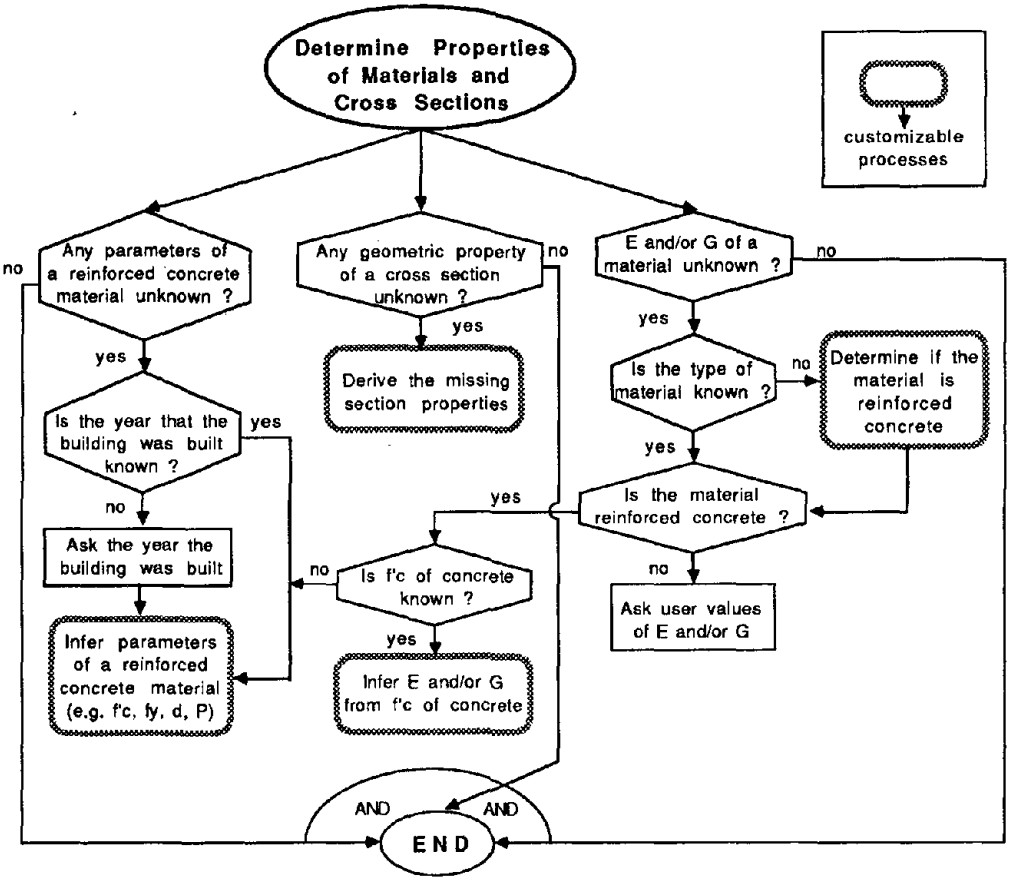


Figure 5-18: Process of determining the properties of materials and cross sections

The process of evaluating a HS starts by classifying the materials of the slabs that form the HS as *rigid* or *flexible*. Slabs made of *rigid* materials are assumed to be infinitely rigid and slabs made of *flexible* materials are assumed to have no in-plane stiffness. Classification is done by the name of the material. If the material of a given slab has already been classified, the classification process is not repeated. The classification criteria are controlled via a knowledge source.

If the HS contains more than one slab, the connections between the slabs of the HS are reviewed. The purpose of this evaluation is to determine whether a group of individual interconnected slabs can behave as a single rigid unit.

A monolithic slab in a building can have a concave shape. If this is the case, the user must break this slab into several convex slabs in order to provide valid input to EVAL. In this case, the group of slabs (if rigid) should be considered to act as a single rigid body. On the other hand, there may be cases in which the connection between two rigid slabs can not be considered adequate enough to make the slabs behave as a single rigid unit. Finally, there may be cases where two slabs made of different materials or

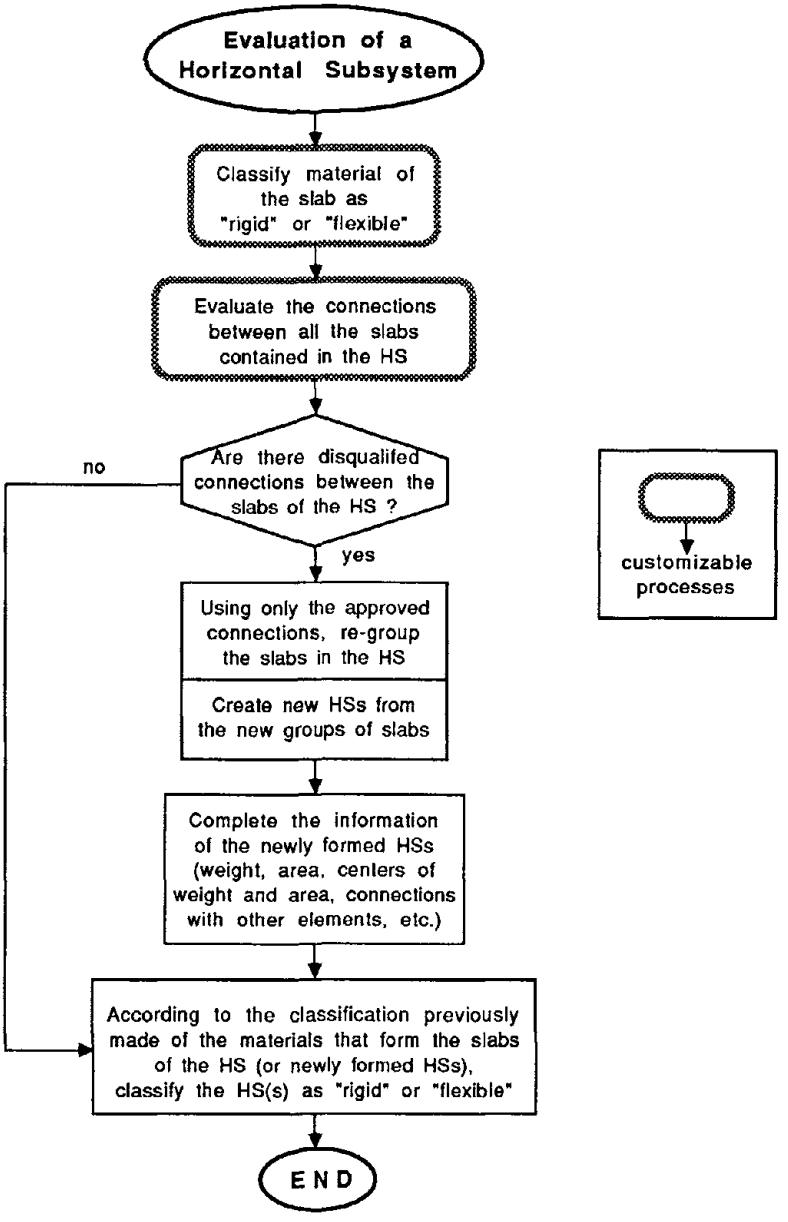


Figure 5-19: Evaluation process of a horizontal subsystem

two slabs with different thicknesses are contiguous. In this evaluation task, the connections between slabs are approved or disqualified. If the connection between two slabs is disqualified, it will be assumed that the slabs are not rigidly connected.

A knowledge source provides the criteria for approving or disqualifying connections between contiguous slabs, based on the thicknesses of the connected slabs and their materials.

After reviewing all the connections between slabs in a HS, if there are any disqualified connections, the slabs are regrouped considering only the approved connections. Depending on the number of disapproved connections between slabs, a HS may be split into several subsystems, as shown in Figure 5-20. If a HS is decomposed into several new ones, information about the newly formed HSs is recomputed.

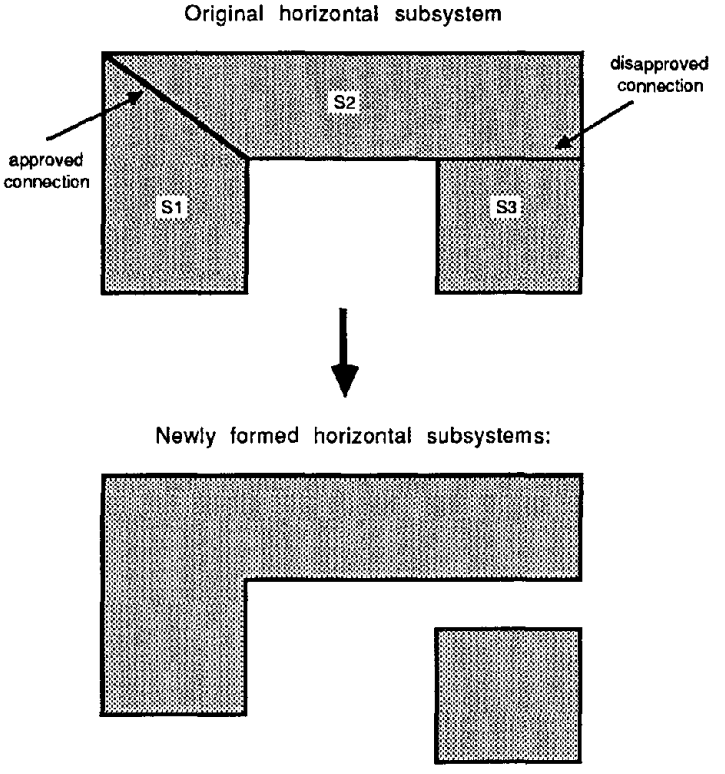


Figure 5-20: Decomposition of a horizontal subsystem

Once all the materials of the slabs have been labeled as *rigid* or *flexible*, the original HS, or the newly formed HSs, are labeled as *rigid* or *flexible* according to the following simple criterion:

- if all the slabs in a HS are flexible, the HS is classified as *flexible*;
- otherwise, the HS is classified as *rigid*.

5.6.5 Structural Analysis of the Building

This section describes the model created by EVAL for structural analysis, including the computation of the seismic forces. The tasks of modeling and analysis are not subject to customization.

Model of the Building. This section presents the idealization of the building and the different behavior models for analysis.

The building is idealized as a series of interconnected nodes forming a network. The connectors between nodes are the individual elements. This network idealization is then translated into the stiffness matrix used to compute the response of the model to the seismic forces in terms of:

- displacements of the nodes of the building; and
- internal forces developed in the individual elements.

EVAL considers two models for analysis: three dimensional and two dimensional models. In the three dimensional model, EVAL uses the *tier building model*, with three degrees of freedom (DOF) per structural node:

- translation in the X direction;
- translation in the Y direction; and
- rotation about the Z axis.

Figure 5-21 shows the DOF of a structural node according to the tier building model.

In the two dimensional model, the building is analyzed separately in two orthogonal directions (X,Y) and in each direction only one DOF per node is considered: translation in the current direction (X or Y) without rotation.

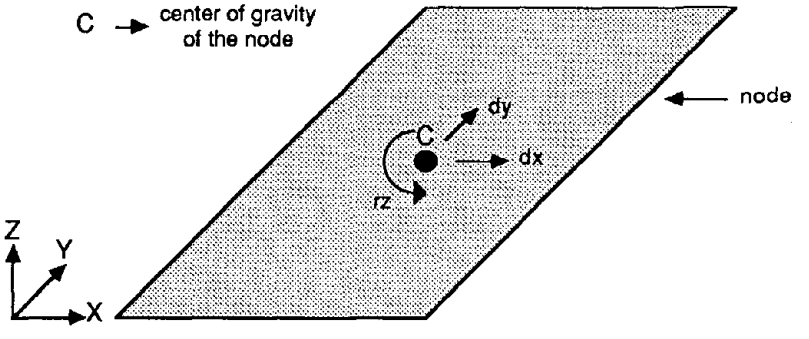


Figure 5-21: Nodal degrees of freedom considered in the tier building model

The following are considered structural nodes:

- horizontal subsystems previously classified as rigid;
- connections between linear elements not contained inside a rigid horizontal subsystem;
- free-standing walls or walls not completely contained inside a rigid horizontal subsystem; and
- connections between individual linear or planar elements and the ground.

The walls are treated as wide columns and the effects of shear deformation are included. Figure 5-22 illustrates examples of structural nodes. There can be three types of structural node:

- *two-dimensional* nodes (D-2); corresponding to rigid horizontal subsystems formed by one or more individual slabs;
- *one-dimensional* nodes (D-1); corresponding to sections of walls treated as structural nodes; and
- *zero-dimensional* nodes (D-0); corresponding to intersections between two linear elements or a linear element and the ground.

D-2 nodes are considered as infinitely rigid bodies. Every horizontal subsystem corresponds to a D-2 node and only slabs made of materials previously classified as *rigid* can belong to horizontal subsystems. A slab made of a *flexible* material is assumed to provide no in-plane stiffness; however, it has a weight and this weight generates seismic forces acting on the center of gravity of the slab. These seismic forces are distributed among the elements that support the flexible slab. Similarly, D-1 nodes are assumed to be inextensible, infinitely rigid, linear segments.

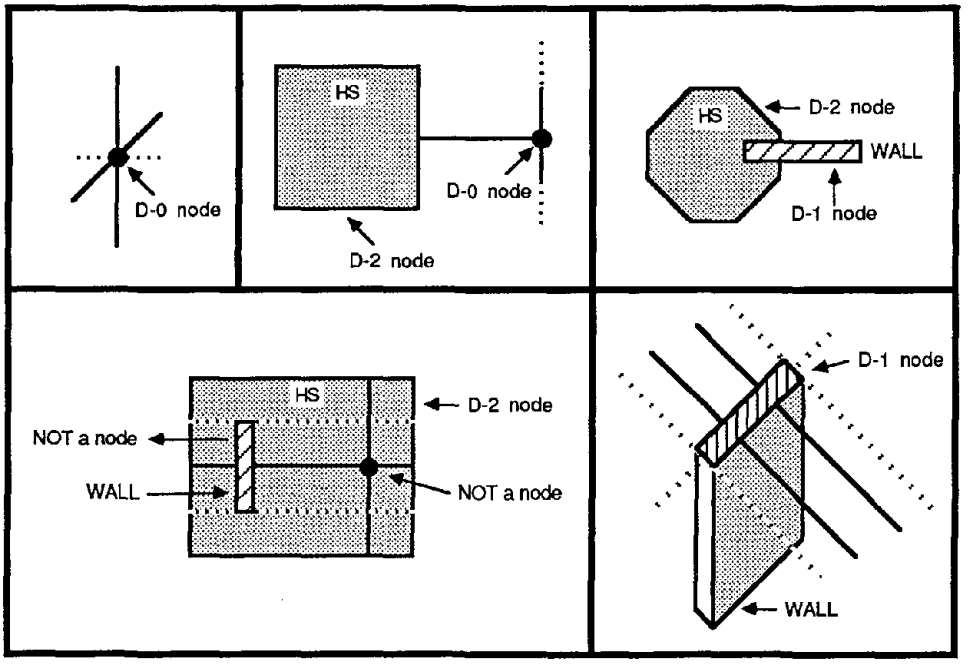


Figure 5-22: Examples of structural nodes

Constraints. As mentioned, the analysis model considers two types of rigid bodies: D-2 and D-1 nodes. In case that a rigid body is connected to one or more other rigid bodies, the displacements of the connected rigid bodies are coupled, imposing constraints on the displacements. EVAL identifies cases in which rigid bodies are connected to generate the appropriate constraints on the displacements. There are three general cases of constraints:

- contact between a horizontal subsystem and a wall;

- contact between two horizontal subsystems; and
- contact between two walls.

In the treatment of these constraints, presented below, small rotations are assumed.

Figure 5-23 illustrates the case of a wall in contact with a horizontal subsystem. In the figure, both the horizontal subsystem (HS) and the wall are structural nodes and each of them has its own set of displacements (dx,dy,rz). However, since the two rigid bodies are connected, their displacements are coupled. In the three dimensional case, two constraints are generated. The first constraint (C1) specifies that the rotation of both rigid bodies is the same. The second constraint (C2) states that the displacement of both rigid bodies along the axis of the wall is the same. The constraints in the case of the two dimensional analysis enforce the displacements of both rigid bodies to be the same along the axis of the wall.

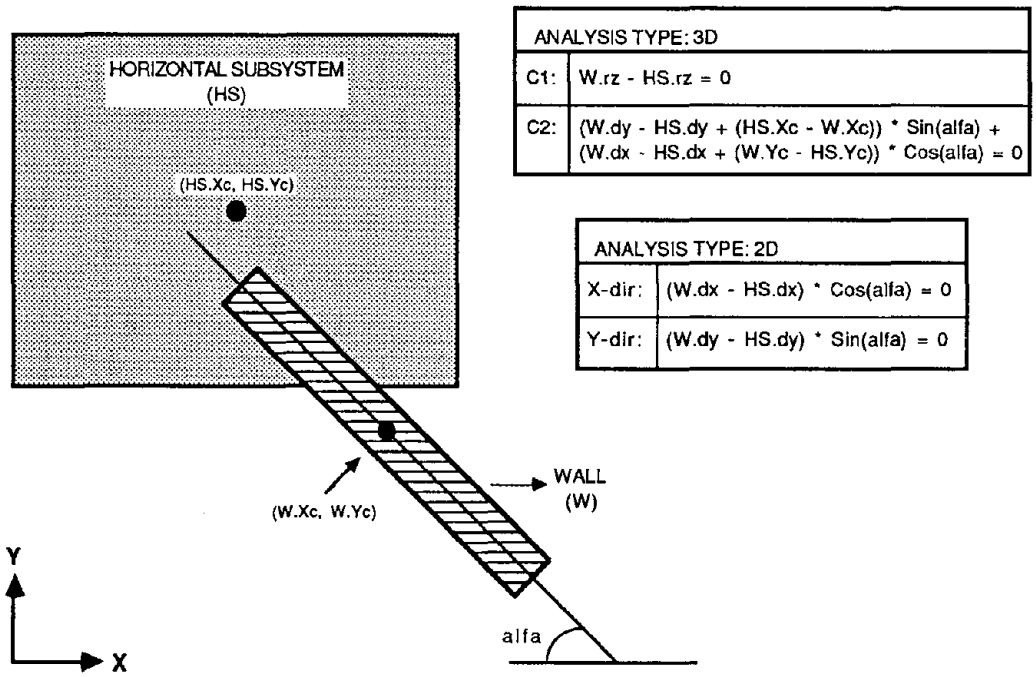


Figure 5-23: Coupling of the displacements of a horizontal subsystem and a wall

Figure 5-24 illustrates two HSs in contact; both HSs are individual structural nodes. In the three dimensional case, three constraints are generated. The first constraint (C1) equates the rotation of the two HSs, and the second and third constraints (C2, C3) state that the displacements (dx,dy) of both HSs must coincide with the simultaneous center rotation of the two HSs. In the two dimensional case, rotations are not considered and the constraints state that the displacements of both HSs must be the same in the X or Y directions.

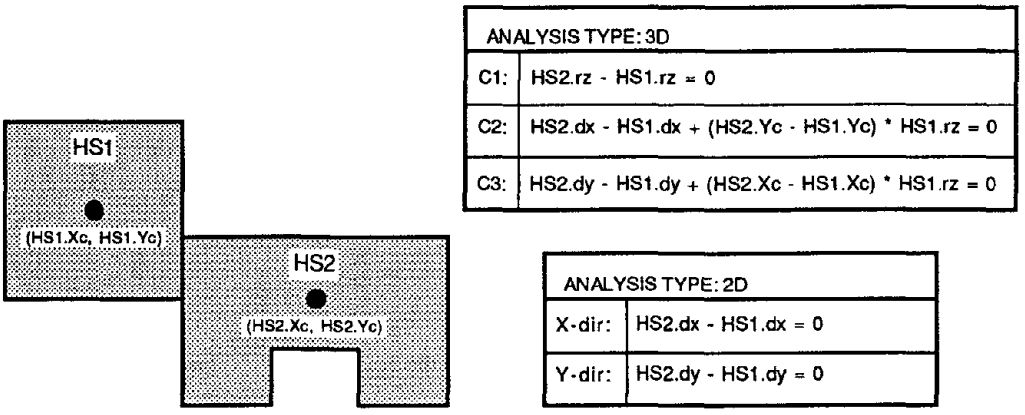


Figure 5-24: Coupling of the displacements of two horizontal subsystems

The last case of two rigid bodies in contact is when two walls touch, as shown in Figure 5-25. The constraints for both types of analysis, two and three dimensional, are the same and state that the displacements of both walls along their axis must be the same. This constraint represents the inextensibility of the walls along their axes.

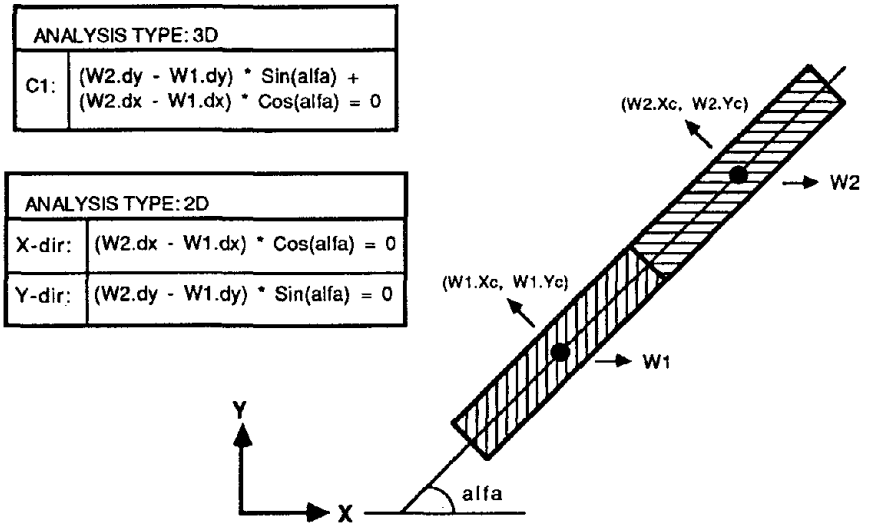


Figure 5-25: Coupling of the displacements of two walls

Determination of Seismic Forces. The seismic forces are considered static horizontal loads acting on the centers of gravity of all the masses. It is assumed that these masses are the slabs; slabs have their own weight and can carry superimposed gravity loads. Gravity loads are transferred from the slabs to the elements that support the slabs and finally to the ground. Two sets of seismic forces are computed: one for the X direction and one for the Y direction.

The first step for computing the seismic forces is to consider the superimposed gravity loads in the building, if any. Once the superimposed gravity loads have been incorporated, the sum of all the weights of the slabs in each floor is computed. The seismic forces acting at each floor level of the building in both the X and Y direction are computed according to the model shown in Figure 5-26, where *csx* and *csy* are the seismic coefficients in the X and Y direction respectively. These coefficients are provided by the user

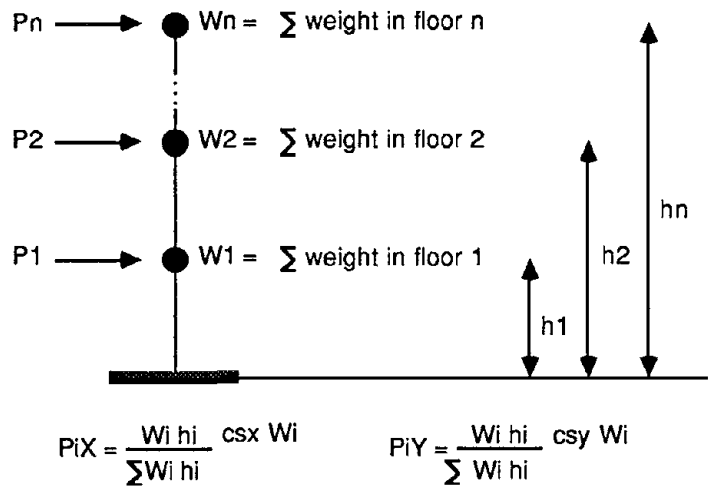


Figure 5-26: Model for computing the seismic forces

in the input file. The forces *PiX* and *PiY* represent the total seismic forces acting on floor *i* in the X and Y direction, respectively. The total seismic force at each floor level is distributed among the horizontal subsystems and flexible slabs at that floor level, proportional to the weight of each element.

Structural Analysis Task. The structural analysis task consists of computing the response of the building to the seismic forces. The structural analysis task is summarized in Figure 5-27. As can be seen in Figure 5-27, the structure is analyzed twice if 2D analysis is requested: first in the X direction and then in the Y direction.

After identifying the DOFs in the building, the equations of the constraints on the displacements of the nodes are entered into the constraint matrix G. The position of the coefficients of the constraint equations within the matrix G is dictated by the DOF numbers associated with the constrained displacements. Matrix G is included with the stiffness matrix of the building, as shown in Figure 5-28a.

The stiffness matrix is symmetric and sparse. To reduce storage, EVAL takes advantage of these properties. The stiffness matrix of the building, K, is stored in column skyline format as shown in Figure 5-28b. Before assembling the stiffness matrix, the shape and size of the skyline of K must be determined by reviewing the DOF numbers of the end nodes of all elements.

Once the skyline of K has been defined, the matrix K is assembled. The process of assembling K consists of accumulating all the stiffness matrices of the individual elements using the direct assembly

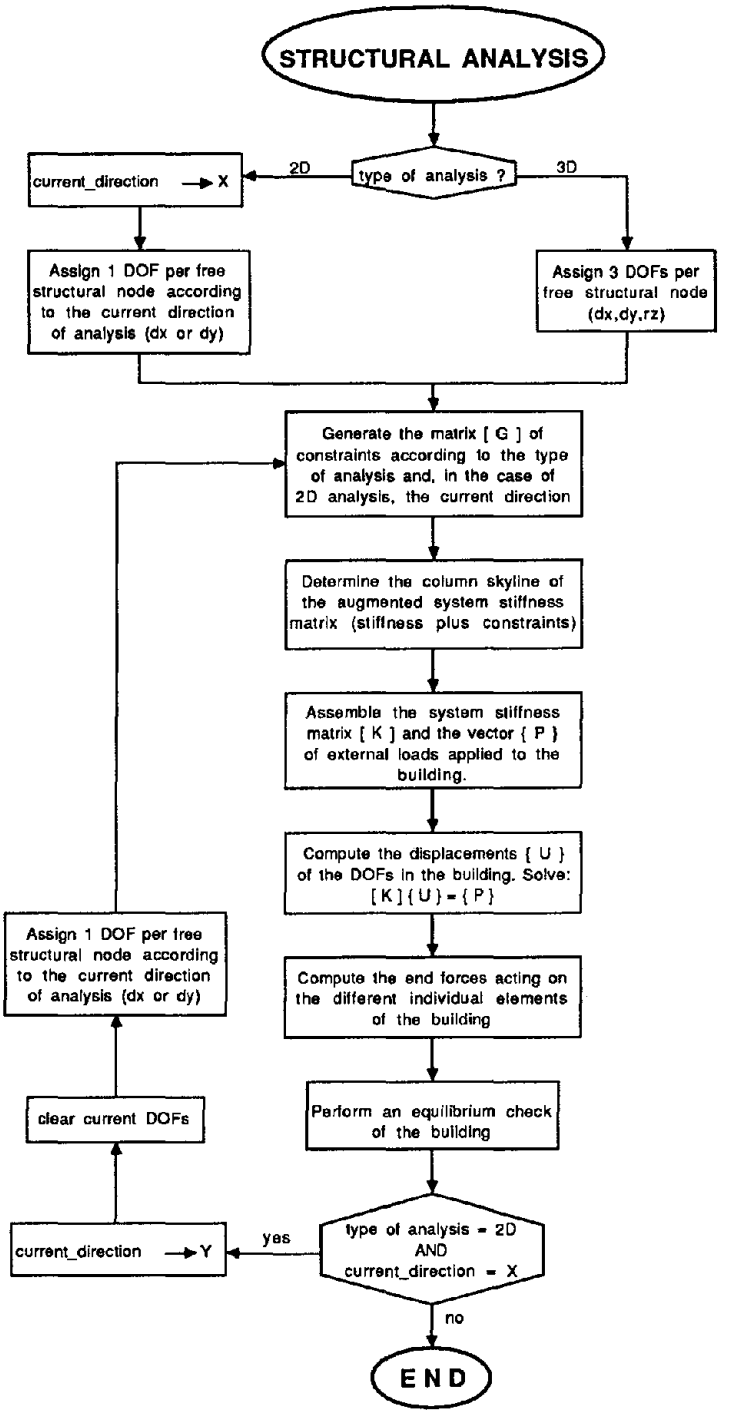


Figure 5-27: Summary of the structural analysis process

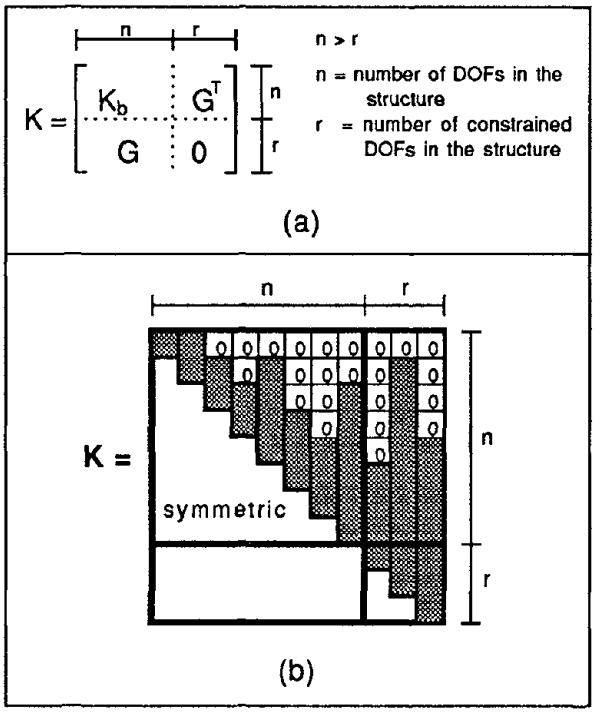


Figure 5-28: Stiffness matrix of the building; skyline representation of the stiffness matrix

method. The process of computing the stiffness matrix of an individual element is illustrated in Figure 5-29 and detailed below.

The computation of an element stiffness matrix starts by computing the stiffness matrix of the current element with respect its local coordinate system (see Figure 5-4). Figure 5-30 shows the stiffness matrix, in the local coordinate system, of an individual element of a space frame. The stiffness matrix in Figure 5-30 includes the effects of shear deformation; these effects are included because walls are idealized as wide columns. In Figure 5-30, *dxA*, *dyA*, and *dzA* are the displacements of the first end of the element, node A, along the X, Y and Z local axes respectively; similarly, *rxA*, *ryA* and *rzA* are the rotations of node A around the X, Y and Z local axes respectively. The same displacements are considered at the second end of the element, node B. The origin of the local coordinate system of the element is considered to be at node A.

The stiffness matrix of the element is rotated from the local coordinate system of the element to the global coordinate system by the following matrix operation:

$$[\, k_{glob} \,] = [\, Rot \,]^T [\, k_{loc} \,] [\, Rot \,].$$

The rotation matrix, *Rot*, used in the transformation operation is shown in Figure 5-31 for an element of a space frame, that is, an element with six degrees of freedom per node. Figure 5-31a illustrates the rotation matrix for an element arbitrarily oriented in space; the terms *Cx*, *Cy* and *Cz* are the direction

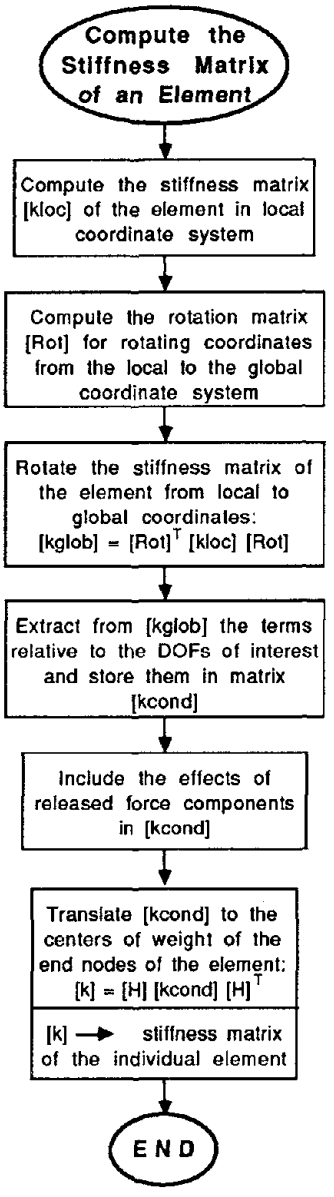


Figure 5-29: Procedure for computing the stiffness matrix of an individual element

cosines of the axis of the element and *alfa* is the angle defined in Figure 5-5. The rotation matrix in Figure 5-31a is valid for any orientation of the element, except when the axis of the element is vertical. The rotation matrix for vertical members is shown in Figure 5-31b.

The 12x12 stiffness matrix of the element in global coordinates, k_{glob}, considers six DOFs per node. The model of behavior assumed in EVAL considers three DOFs per node in the 3D analysis mode and one DOF per node in the 2D analysis mode. The contents of the 12x12 matrix k_{glob} relevant to the DOFs considered in the behavior model assumed by EVAL are extracted and stored in a new, smaller matrix,

dxA	dyA	dzA	rxA	ryA	rzA	dxB	dyB	dzB	rxB	ryB	rzB	
$\frac{EA}{L}$	0	0	0	0	0	$-\frac{EA}{L}$	0	0	0	0	0	dxA
0	$\frac{12\,E\,Iz}{L^3\,(1+PhiY)}$	0	0	0	$\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	0	$-\frac{12\,E\,Iz}{L^3\,(1+PhiY)}$	0	0	0	$\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	dyA
0	0	$\frac{12\,E\,Iy}{L^3\,(1+PhiZ)}$	0	$-\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	0	0	$-\frac{12\,E\,Iy}{L^3\,(1+PhiZ)}$	0	$-\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	dzA
0	0	0	$\frac{GJ}{L}$	0	0	0	0	0	$-\frac{GJ}{L}$	0	0	rxA
0	0	$-\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	$\frac{E\,Iy\,(4+PhiZ)}{L\,(1+PhiZ)}$	0	0	0	$\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	$\frac{E\,Iy\,(2-PhiZ)}{L\,(1+PhiZ)}$	0	rya
0	$\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	0	0	0	$\frac{E\,Iz\,(4+PhiY)}{L\,(1+PhiY)}$	0	$-\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	0	0	0	$\frac{E\,Iz\,(2-PhiY)}{L\,(1+PhiY)}$	rzA
$-\frac{EA}{L}$	0	0	0	0	0	$\frac{EA}{L}$	0	0	0	0	0	dxB
0	$-\frac{12\,E\,Iz}{L^3\,(1+PhiY)}$	0	0	0	$-\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	0	$\frac{12\,E\,Iz}{L^3\,(1+PhiY)}$	0	0	0	$-\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	dyB
0	0	$-\frac{12\,E\,Iy}{L^3\,(1+PhiZ)}$	0	$\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	0	0	$\frac{12\,E\,Iy}{L^3\,(1+PhiZ)}$	0	$\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	dzB
0	0	0	$-\frac{GJ}{L}$	0	0	0	0	0	$\frac{GJ}{L}$	0	0	rxB
0	0	$-\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	$\frac{E\,Iy\,(2-PhiZ)}{L\,(1+PhiZ)}$	0	0	0	$\frac{6\,E\,Iy}{L^2\,(1+PhiZ)}$	0	$\frac{E\,Iy\,(4+PhiZ)}{L\,(1+PhiZ)}$	0	ryB
0	$\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	0	0	0	$\frac{E\,Iz\,(2-PhiY)}{L\,(1+PhiY)}$	0	$-\frac{6\,E\,Iz}{L^2\,(1+PhiY)}$	0	0	0	$\frac{E\,Iz\,(4+PhiY)}{L\,(1+PhiY)}$	rzB

L ⟶ Longitudinal length of the element

A ⟶ Area of the cross section of the element

J ⟶ torsional constant of the cross section

Iy, Iz ⟶ moments of inertia of the cross section with respect the local Y and Z axes

PhiY ⟶ $24\ Fs\ (1 + V)\ (ry / L)^2$

PhiZ ⟶ $24\ Fs\ (1 + V)\ (rz / L)^2$

Fs ⟶ Shape factor of the cross section; Fs = A / Shear_Area

Shear_area ⟶ Area where the shear stress is supposed to be acting

r y ⟶ radius of giration of the cross section in the local Y direction: $ry = \sqrt{Iy / A}$

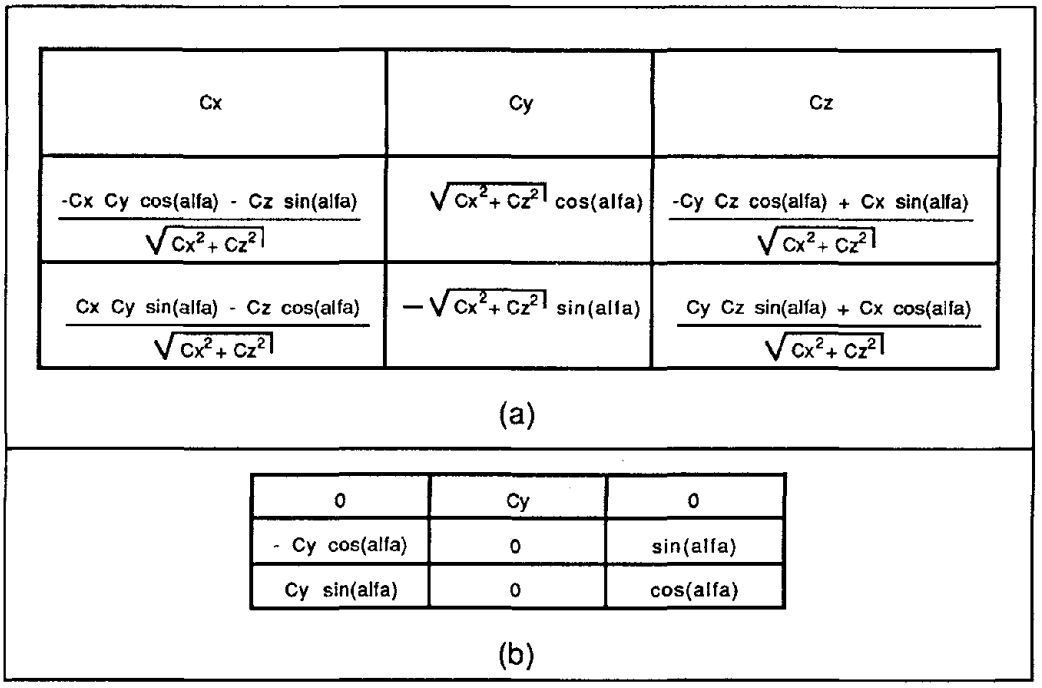
r z ⟶ radius of giration of the cross section in the local Z direction: $rz = \sqrt{Iz / A}$

V ⟶ Poisson ratio of the material

E ⟶ Young's modulus of the material

G ⟶ Shear modulus of the material

Figure 5-30: Stiffness matrix in local coordinate system of an individual element in space



Cx	Cy	Cz
$\frac{-Cx\ Cy\ \cos(alfa) - Cz\ \sin(alfa)}{\sqrt{Cx^2 + Cz^2}}$	$\sqrt{Cx^2 + Cz^2}\ \cos(alfa)$	$\frac{-Cy\ Cz\ \cos(alfa) + Cx\ \sin(alfa)}{\sqrt{Cx^2 + Cz^2}}$
$\frac{Cx\ Cy\ \sin(alfa) - Cz\ \cos(alfa)}{\sqrt{Cx^2 + Cz^2}}$	$-\sqrt{Cx^2 + Cz^2}\ \sin(alfa)$	$\frac{Cy\ Cz\ \sin(alfa) + Cx\ \cos(alfa)}{\sqrt{Cx^2 + Cz^2}}$

(a)

0	Cy	0
- Cy cos(alfa)	0	sin(alfa)
Cy sin(alfa)	0	cos(alfa)

(b)

Figure 5-31: Rotation matrices for elements of three-dimensional frames

k_{cond}

The force components in an element associated with the DOFs of its end nodes might be released either by the user or as result of the seismic evaluation process. If an individual element contains released force components, the effects of these releases must be incorporated into the stiffness matrix of the element. The procedure for including the effects of releases is presented in [37] and involves a series of matrix operations and condensations. The reason for shrinking k_{glob} into k_{cond} is to perform the matrix operations on a smaller matrix, increasing the speed of the process.

At this point, the stiffness matrix of the element has been:

- rotated into the coordinate system of the building;
- reduced to the DOFs of interest, and
- modified to reflect the effects of possible releases of force components of the element.

If the two end nodes of the element are D-0 nodes, the process of computing the stiffness matrix of the element has been completed. However, if either of the two end nodes of the element is not a D-0 node, the stiffness matrix of the element has to be translated to relate the displacements of the centers of gravity of the two end nodes. Figure 5-32 shows an element whose two end nodes are D-2 nodes.

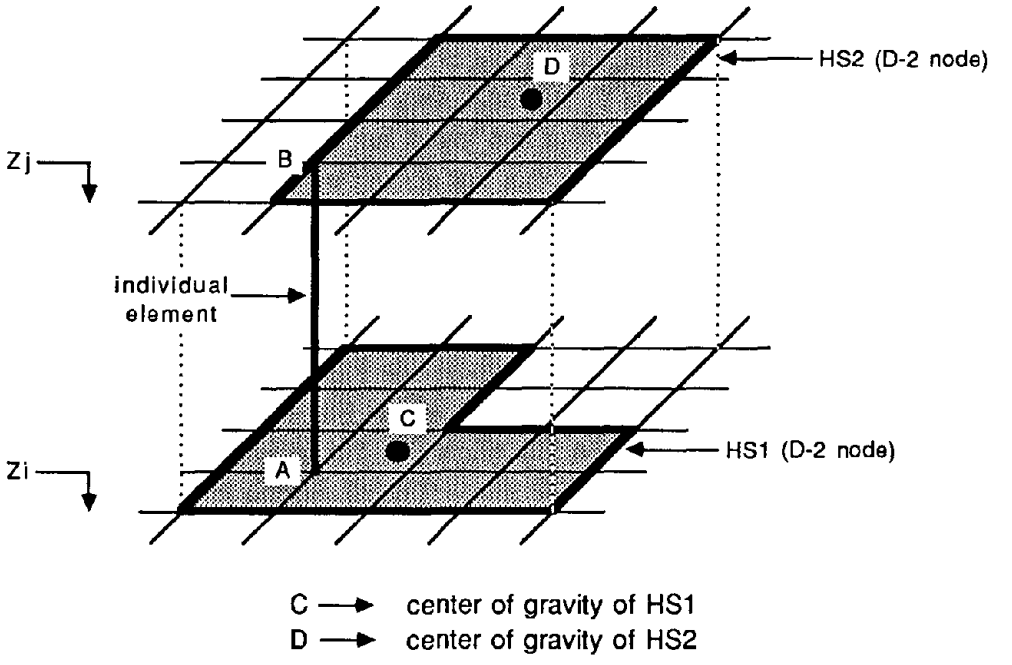


Figure 5-32: Translation of the stiffness of an element to the centers of gravity of the end nodes of the element

Elements are links that join two structural nodes, and the stiffness matrix of an element must relate the displacements of the centers of gravity of both end nodes. In the case of D-2 and D-1 nodes, an element can be in contact with the node at a point different from the center of gravity of the node; if this is the case, the stiffness of the element must be *translated* into the centers of gravity of each node with the procedure summarized in Figure 5-33. In Figure 5-33, matrix H_{ij} represents the translation of a force

$$\{ Px, Py, Mz \}^T$$

from point i to point j.

Once the stiffness matrix, K, and the vector of external loads, P, have been assembled, the following system of simultaneous equations is solved to find the displacements of the DOFs and the interaction forces that insure the constraints between the displacement of rigid bodies connected between them:

$$\begin{bmatrix} K_b & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} U \\ R \end{bmatrix} = \begin{bmatrix} P \\ 0 \end{bmatrix}$$

In the above system of equations, *U* is the vector of displacements of the free nodes and *R* is the vector of interaction forces that insure the constraints between the displacement of rigid bodies connected between them (Lagrange multipliers). The system of equations is solved using a modified version of the Cholesky method [26].

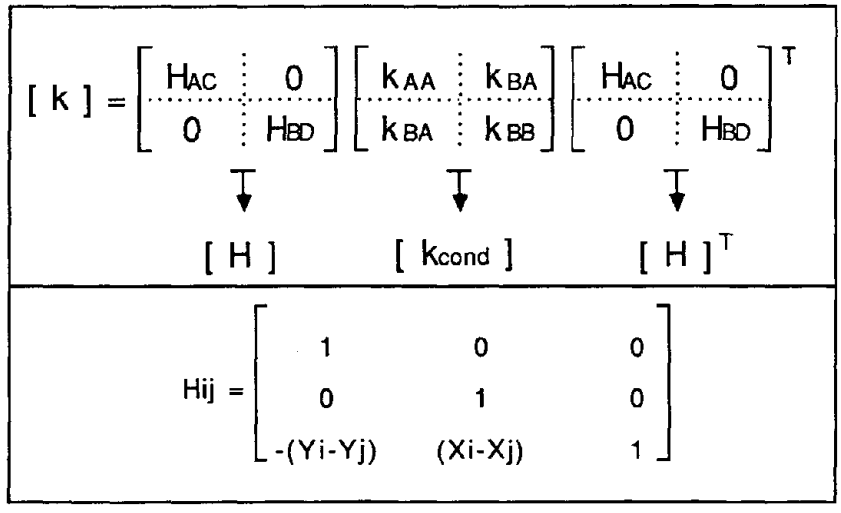


Figure 5-33: Procedure for translating the stiffness of an element from the end points of the element (A,B) to two arbitrary points (C,D)

If matrix K is found to be singular, the building is either unstable or overconstrained. EVAL aborts the execution of the system if the building is unstable and issues a warning message. The reason of singularity is pointed out by EVAL: instability or overconstraint. If the building is unstable, EVAL reports the first node that was found to be unstable and identifies the particular DOF of that node that caused singularity.

The displacements induce internal forces in the individual elements. These forces are computed by multiplying the stiffness matrix of the element by the displacements of the end nodes. Since the displacements of the end nodes of an element refer to the center of gravity of these nodes, the displacements of the end nodes have to be translated to the points where the element is in contact with the end nodes. For computing the end forces of the element AB shown in Figure 5-32, the displacements of node HS2 have to be translated from point D to point B, and the displacements of HS1 have to be translated from point C to point A. A displacement vector of the type

$$\{ dx, dy, rz \}^T$$

is translated from point i to point j with the following operation:

$$\{ d_i \} = [H_{ij}]^T \{ d_j \},$$

where H_{ij} is the matrix shown in Figure 5-33.

To insure that the results of the analysis are correct, the equilibrium of each free node is checked. The process of verifying equilibrium is done simultaneously with computing the end forces of individual elements; this process also leads to the computation of the reactions at the supports.

The end forces of an element are rotated, using the rotation matrices of Figure 5-31, to the global

coordinate system of the building, and translated, using matrix H, to the centers of gravity of the end nodes and accumulated. The interaction forces R are also accumulated at the nodes that are coupled between them. If the result of the accumulation of forces in the center of gravity of a free node is equal to the external forces applied to that node, then the node is in equilibrium. A building is in equilibrium only if all the nodes of the building are in equilibrium. In the case of restrained nodes (supports), this procedure of accumulation of forces leads directly to the reactions in the global coordinate system.

5.6.6 Determination of Redundancy in the Transmission of Seismic Loads

This task, summarized in Figure 5-34, evaluates the way in which the seismic loads are transmitted from a horizontal subsystem to the elements that support it, to determine whether there is sufficient redundancy in the transmission of seismic loads.

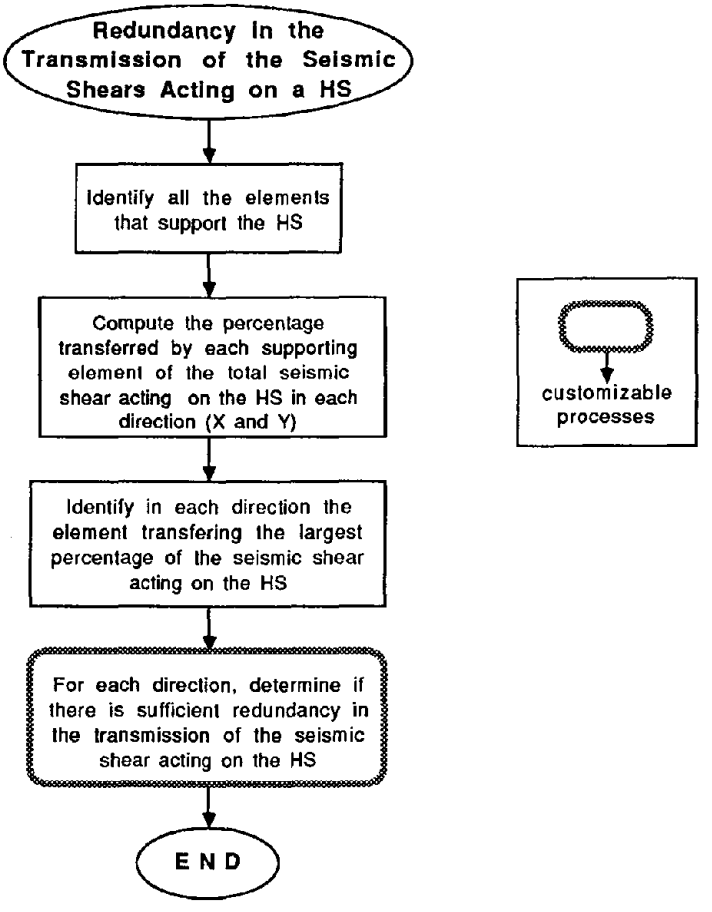


Figure 5-34: Determination of redundancy in the transmission of the seismic shears acting on a horizontal subsystem

A horizontal subsystem should not depend significantly on a small number of structural elements to transfer the seismic loads acting on the subsystem to the rest of the building and later to the ground.

Such dependency will jeopardize the stability of the horizontal subsystem in case that the relevant structural elements fail. However, the definition of redundancy in the transmission of lateral loads is highly subjective. This section describes the procedure for computing the ingredients for assembling a definition of redundancy in the transmission of lateral loads. The default definition of redundancy in EVAL is presented in the next section.

This task requires a previous two-dimensional analysis; the reason for specifically requiring the 2D analysis will be justified below. Once the analysis has been performed, the individual elements that transmit the seismic forces acting on the horizontal subsystems are identified and a load percentage is assigned to them. This load percentage corresponds to the ratio of the force carried by the element in a particular direction (X or Y) to the total seismic shear acting on the horizontal subsystem in that direction. Once these percentages have been assigned, the maximum percentages on each direction are identified and the number of elements carrying the seismic forces of the horizontal subsystem are counted; the elements with released force components are not taken into account. The above procedure is summarized in Figure 5-35.

The reason for using the results of a 2D structural analysis is that in this type of analysis all the end forces of the individual supporting elements, in the two orthogonal directions, have the same orientation. If a 3D analysis were to be performed, the end forces of elements supporting the same horizontal subsystem may have different orientations due to the effects of torsion; this causes negative percentages in the table shown in Figure 5-35 that would not be meaningful in the determination of redundancy.

5.6.7 Evaluation of Torsion in Horizontal Subsystems

This task, summarized in Figure 5-36 has two main objectives:

- determine whether a horizontal subsystem is subject to excessive torsion; and
- identify the elements that make a significant contribution to the torsional stability of a horizontal subsystem; these elements are pointed out since they are critical to the torsional stability of the horizontal subsystem.

This task requires a previous three-dimensional analysis. Once the building has been analyzed, the center of torsion (CT) of each horizontal subsystem is computed according to the following formulas.

$$Xt = \frac{\Sigma Fyi\ Xi}{\Sigma Fyi}$$

$$Yt = \frac{\Sigma Fxi\ Yi}{\Sigma Fxi}$$

Fxi and *Fyi* are the forces, in the global coordinate system, acting on the ends of the elements that support the horizontal subsystem. Only the elements that support the horizontal subsystem are considered in the above formulas. Beams completely contained within a horizontal subsystem are not

HORIZONTAL SUBSYSTEM				
Seismic shears acting on the center of gravity of the HS Px = 12 Py = 20				
Elements supporting the HS:				
ID	Fx	Fy	% Px	% Py
Col 1	3	6	25.0	30.0
Col 2*	0	3	0.0	15.0
Col 3	2	5	16.7	25.0
Col 4	7	6	58.3	30.0
	Σ= 12	Σ= 20	Σ= 100	Σ= 100
Max % Px = 58.3		Max % Py = 30.0		
Number of elements that transmit the seismic shears: In the X direction = 3 In the Y direction = 4				
* Fx component previously released				

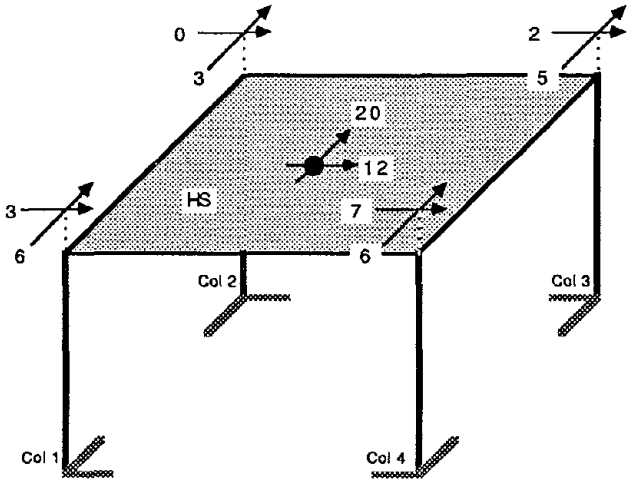


Figure 5-35: Information for defining redundancy in the transmission of seismic loads

taken into account since they are self-equilibrating. *Xi* and *Yi* are the coordinates of the points where the elements are in contact with the horizontal subsystem. If the center of torsion is significantly far away from the center of gravity of the horizontal subsystem, the subsystem is subject to considerable torsion. The diagnosis of a significant torsion problem is controlled with a knowledge source.

After the centers of torsion of the horizontal subsystems have been computed, EVAL determines the contribution to the torsional stability of a horizontal subsystem made by each of the elements that support

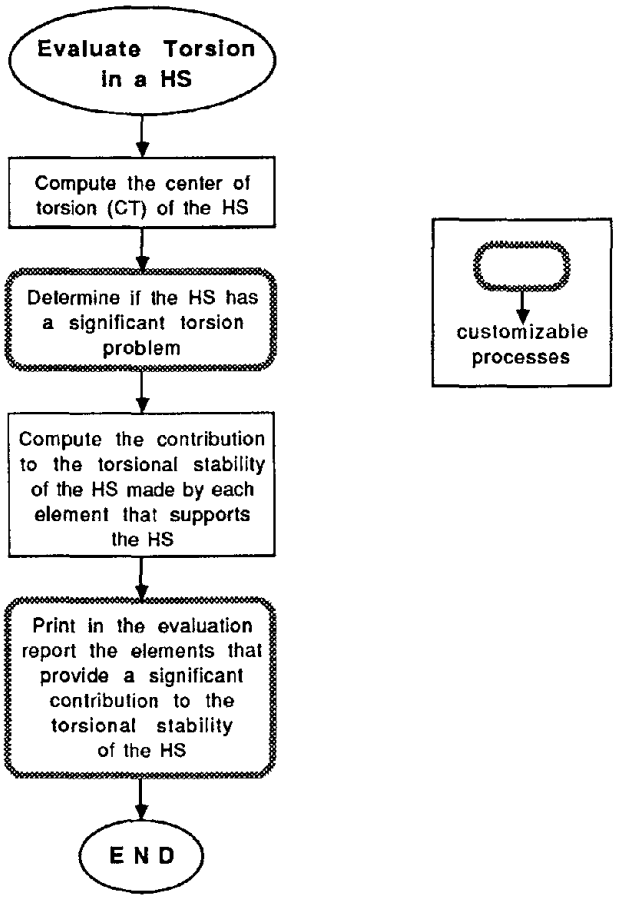
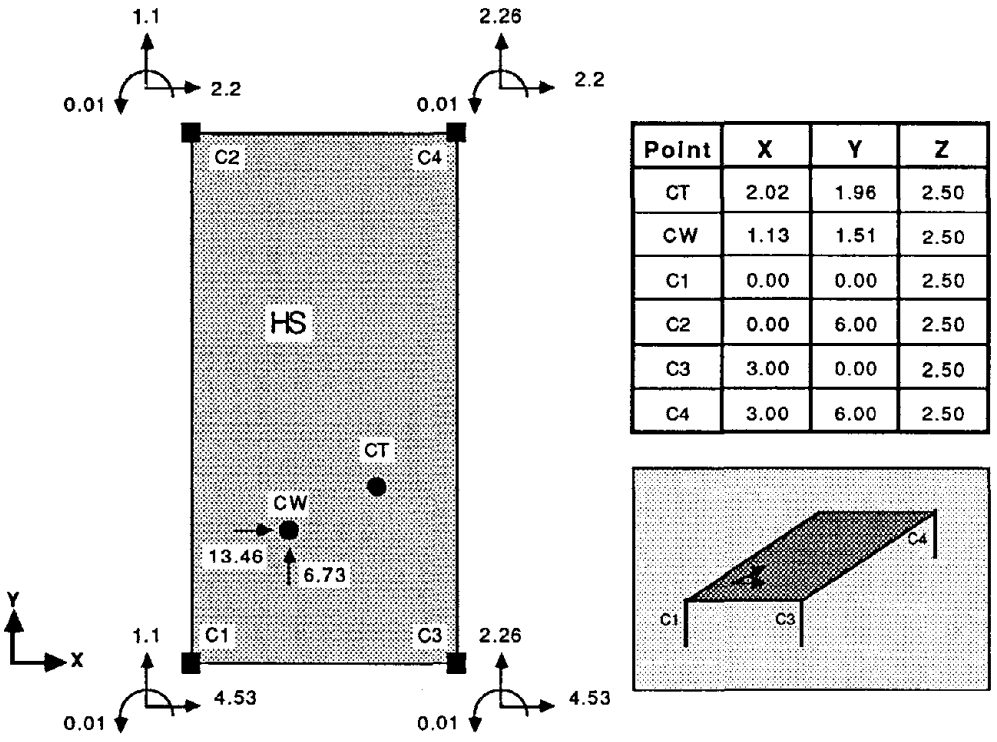


Figure 5-36: Evaluation of torsion in a horizontal subsystem

it. The process of computing the contribution to the torsional stability of a horizontal subsystem made by the elements that support it is illustrated in Figures 5-37 and 5-38.

Figure 5-37 illustrates a simple building consisting of four columns supporting a slab. This structure is subject to two seismic forces of magnitudes 13.46 and 6.73 respectively, acting on the center of gravity of the building along the X and Y directions. The center of gravity (CW) of the horizontal subsystem (the slab) does not coincide with the geometric center of the horizontal subsystem. This eccentricity, due to the symmetry in the location of the columns, causes an unequal distribution of the end forces acting on each of the four columns that support the horizontal subsystem. The values of the end forces acting on each of the columns are shown in Figure 5-37. The values in the figure were used for computing the location of the center of torsion of the horizontal subsystem (CT). The world coordinates of the centers of gravity and torsion are listed in Figure 5-37. Once the center of torsion is known, the sets of end forces of the elements are translated to the center of torsion of the building. This translation of forces, as illustrated in Figure 5-38, is made with matrix H, shown in Figure 5-33.



Point	X	Y	Z
CT	2.02	1.96	2.50
CW	1.13	1.51	2.50
C1	0.00	0.00	2.50
C2	0.00	6.00	2.50
C3	3.00	0.00	2.50
C4	3.00	6.00	2.50

Figure 5-37: Sample building used for illustrating the evaluation of torsion in a horizontal subsystem

The sum of moments of all the elements that support the horizontal subsystem, translated to the center of torsion, must add to zero. Once all the set of forces have been translated into the the center of torsion, the moments of each sign, negative and positive, are accumulated; the result is two identical moments, M_+ and M_-, with opposite signs. The contribution made by each element to the torsional stability of a horizontal subsystem is measured in terms of a percentage, which can be positive or negative. This percentage is computed by dividing the moment component of a set of forces acting on the center of torsion by M_+, as illustrated in Figure 5-38. Elements with a higher contribution percentage associated to them, in absolute value, make a more significant contribution to the torsional stability of the horizontal subsystem than elements with a lower percentage. The elements that provide a significant contribution to the torsional stability of a horizontal subsystem are presented to the user in the evaluation report. A knowledge source controls the threshold that defines what percentage of contribution starts to be relevant.

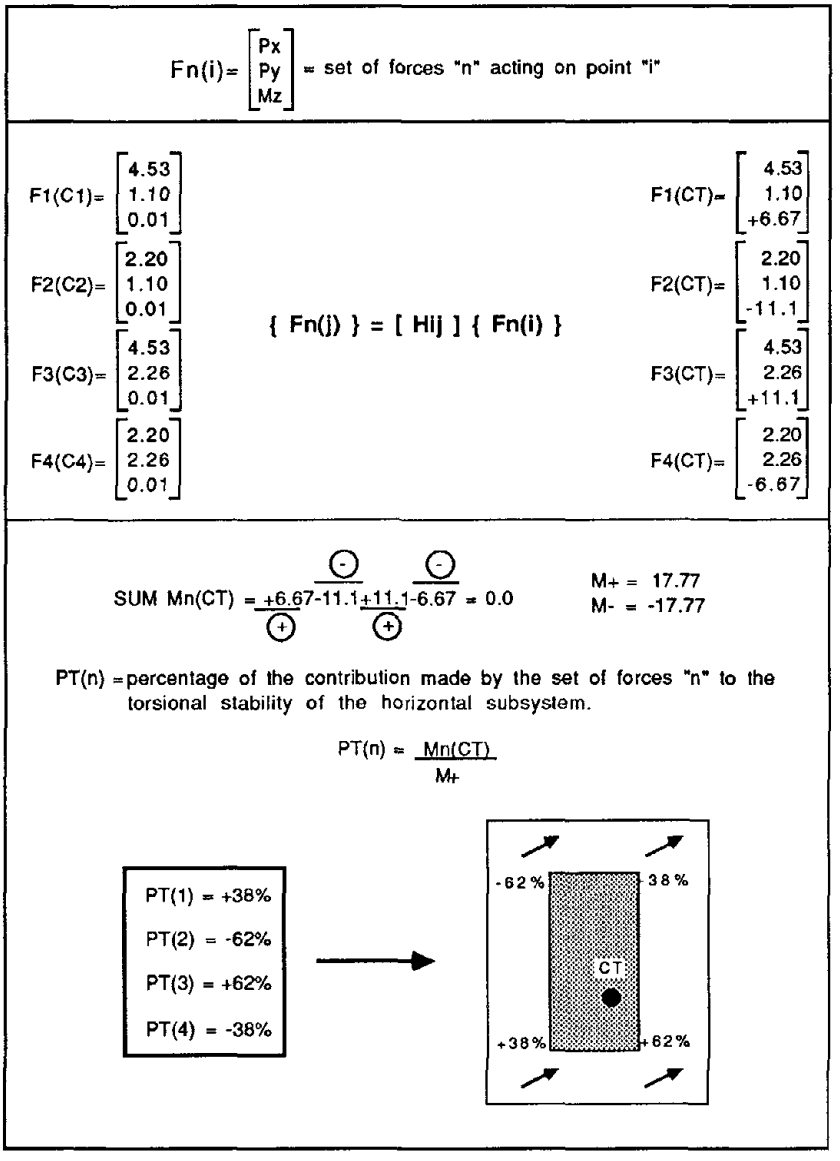


Figure 5-38: Determination of the contribution to the torsional stability of a HS made by each of the elements that support the HS

5.6.8 Evaluation of the Resistance of Individual Elements and Connections

This task consists of determining whether the internal forces acting on individual elements can be resisted by these structural elements. This task is executed after analyzing the building with the 3D analysis model. The 3D model of analysis is used because it is more realistic than the 2D model.

As a result of the structural analysis, each of the local segments of a linear element has a set of forces

at its end nodes (local segments are illustrated in Figure 5-17). The purpose of this task is to check all the connections between elements and *disconnect* or *release* the force components of individual elements if the capacity of these elements for transmitting these force components is exceeded. Figure 5-39 illustrates a connection between several elements. According to the tier building model, each structural element will have three force components at each end of the element. These force components are:

- Px: force acting in the X direction;
- Py: force acting in the Y direction; and
- Mz: moment around the Z axis.

The above moments are referred to the global coordinate system of the building.

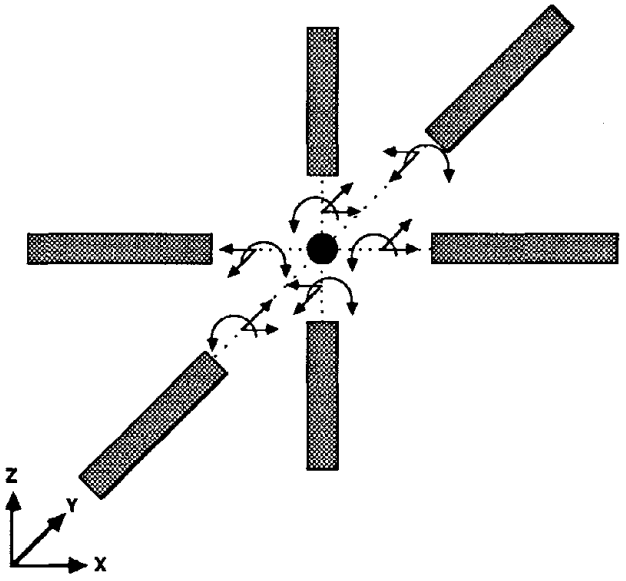


Figure 5-39: End forces of individual elements in a structural connection

A force component of the set (Px,Py,Mz), acting at the end of an element, is released if the element on which the component is acting cannot transmit the magnitude of such force component. In EVAL, the release of a force component can be made only in global coordinates. Depending on the orientation of an element, the forces in the global coordinate system can have different meanings. For example, Mz corresponds to a torsional moment in a column and to a flexural moment in a beam; similarly, Px corresponds to an axial force in a *beam_xx*, to shear in a *beam_yy*, and to axial and shear forces in a *beam_xy*, since Px is decomposed in two orthogonal components relative to the principal axis of the *beam_xy*.

The process of reviewing the elements in a building is done connection by connection. The first step is to locate all the members involved in a a structural connection; after all the elements have been identified, their end forces are evaluated to determine which of these forces will be released. Figure 5-40

summarizes the process followed by EVAL to determine when a force component acting on the end of an individual structural element must be released. The process starts by rotating the set of end forces

$$\{ Px, Py, Mz \}^T$$

from the global coordinate system to the local coordinate system of the element on which the forces are acting on. Since the behavior model assumed in 3D structural analysis considers only translations along the X and Y global direction and rotation around the Z axis, the global components Pz, Mx and My are assigned a value of zero. The purpose of the rotation operation is to have a uniform reference for evaluating the force components acting on an element, and to avoid the different interpretations of these components according to the orientation of the structural element.

The procedure for releasing the force components of a linear element in the local coordinate system is illustrated in Figure 5-41. Since the meaning of each force component in local coordinate system is known and always the same, appropriate judgments are made to determine if each of these components, in local coordinate system, should be released. There are four type of force components acting on a structural element:

- axial force;
- shear force;
- torsional moment; and
- flexural moment.

The user sets the criteria for evaluating when the above force components must be released. The criteria implemented in EVAL for releasing the above force components is presented in the following section. After determining which local force components must be released and releasing them, they are translated back to the global coordinate system.

Since EVAL does not handles releases in local coordinate system, it must be determined which of the force components of interest (Px, Py or Mz), referred to the global coordinate system, should be released. If a structural element is aligned with one of the principal global directions X, Y, or Z (*beam_xx*, *beam_yy* or *column* respectively), it means that a global force component corresponds precisely to one and only one of the local force components. If this is the case, the correspondence between a local release and a global release is precise. However, if a structural element is not aligned with any of the principal directions, the correspondence between a local and a global release is not precise since a global force component corresponds to fractions of several local force components.

To deal with the problem of having to determine global releases from local releases, EVAL follows the procedure described below:

- the vector of force components in global coordinate system, { P }, is rotated to the local coordinate system and stored in vector { P' };
- the forces { P' } in local coordinate system are evaluated to determine which components must be released;

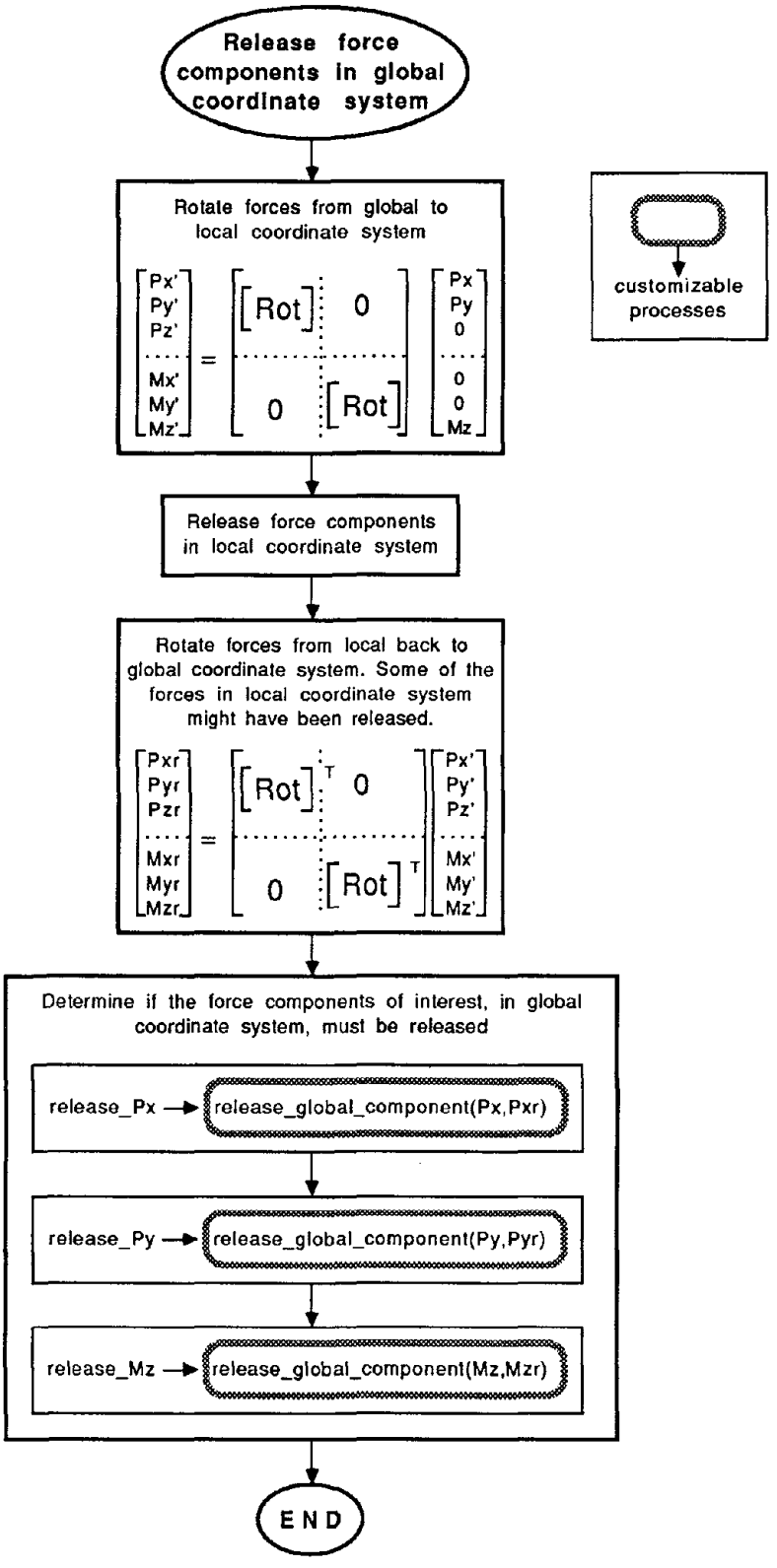


Figure 5-40: Procedure for releasing the relevant force components acting on an element

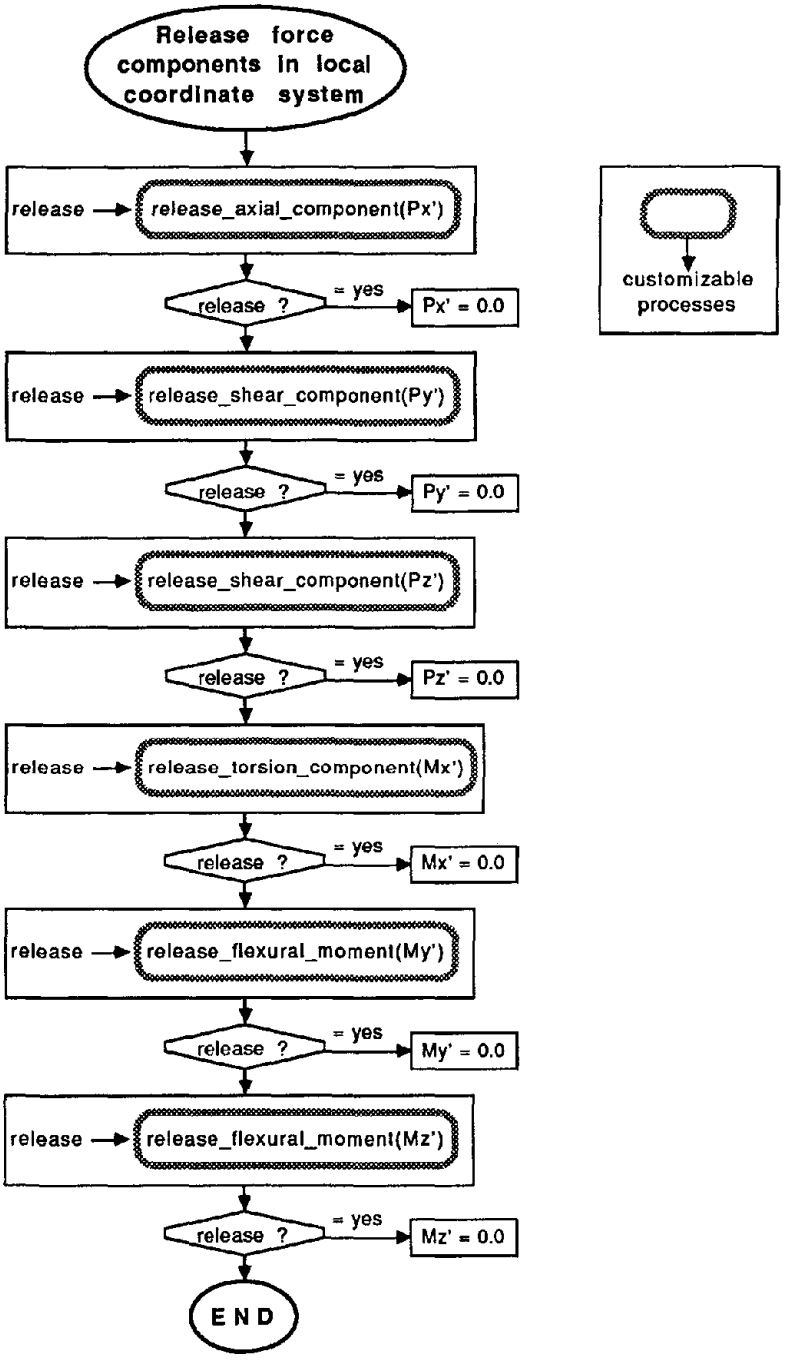


Figure 5-41: Procedure for releasing the force components, in local coordinates system, acting on an element

- { P' }, containing the releases, is rotated back to the global coordinate system and stored in a third vector, { Pr };
- the components of interest in vector { P } and { Pr } are compared: [Px,Prx], [Py,Pry] and [Mz,Mzr], to determine if they should or should not be released.

The user has control on the criteria for comparing these forces; the criteria established for EVAL is presented in the next section.

If a release of a force component is made in any element of the building, EVAL activates the system flag *MADE_A_RELEASE* that indicates that an element has been disconnected in some component from a structural connection. The flag *MADE_A_RELEASE* is used by the control of EVAL to determine whether the building should be re-analyzed to include the effect of a release.

5.6.9 Generation of Evaluation Report

Each of the tasks contained in the core knowledge base generates a report segment with the results or diagnostics produced by the task. The generation of the final evaluation report is a task performed by EVAL after all the evaluations selected by the user have been executed. Each evaluation report segment is represented by a goal. The final evaluation report of EVAL is stored in a text file called *eval-rep* in the current working directory.

SECTION 6
CUSTOM KNOWLEDGE BASE OF EVAL

This section describes an individual customization of the core knowledge base processes, described in the previous section, that are subject to customization. Custom knowledge is represented in decision table format, as described in section 4.

Sources used for defining the heuristics contained in this section.

- relevant literature in the domain of seismic evaluation of existing buildings;
- textbooks of structural mechanics and design;
- previous works developed to infer missing or incomplete information about existing structures; and
- heuristics defined by the authors based on the sources listed above.

6.1 Determination of Material and Cross Section Properties

This section describes the customization of the process of determining the properties of materials and cross sections. As can be seen in Figure 5-18, the process of determining the properties of a material has three sub-processes that are subject to individualization. These sub-processes are illustrated in Figure 6-1. The heuristics for inferring missing parameters of reinforced concrete were taken from [39].

Determination of the Type of Material. It can be seen in Figure 5-18 that if the E or G properties of a material are unknown, and the type of material was not specified, EVAL checks whether the material is reinforced concrete. This sub-task of checking whether the material is reinforced concrete is carried by knowledge source *infer_if_mat_is_concrete*, presented below.

```
*****************************************************************
*                                                               *
* Knowledge Source: infer_if_mat_is_concrete                    *
*                                                               *
* used by: det_mat_rigidity                                     *
*                                                               *
*****************************************************************

Parameter list: material_name -> string
                density -> real
                is_concrete -> string ==> RESULT
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"density" is-a --> density; units: "Ton/m3"
```

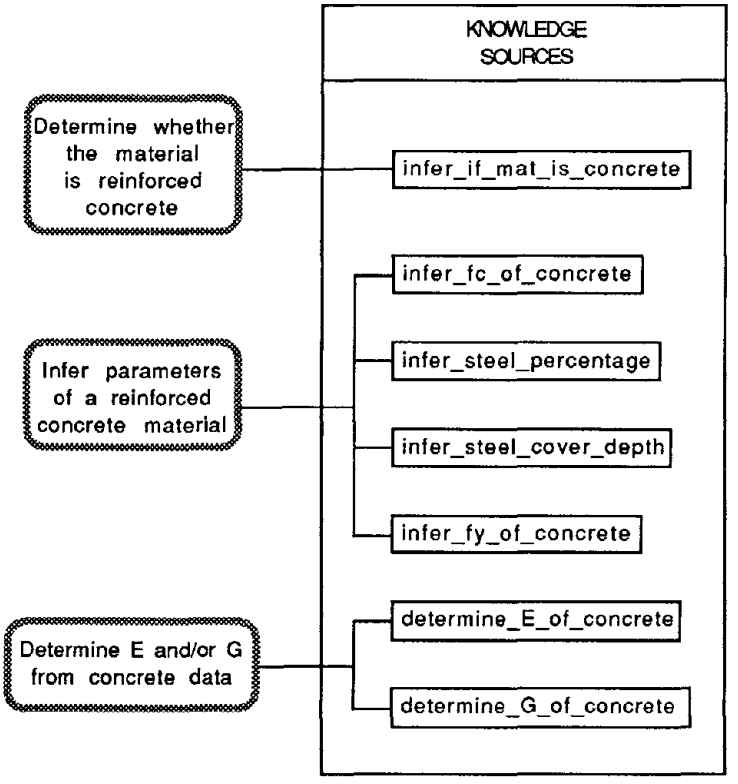


Figure 6-1: Sub-processes subject to individualization in deriving the properties of materials

name_matches_concrete(material_name) != 0	T T	ELSE
density > 1.5	T F	
is_concrete = "yes"	X	
is_concrete = "no"	X	
is_concrete = ask the user		X
explanation = "inferred by EVAL"	X X	
explanation = "provided by the user at run-time"		X

The determination of whether a material corresponds to reinforced concrete is made using the density and name of the material. The default criteria is that if the substring *"concr"* appears somewhere in the name of the material and the density of the material is greater than 1.5 metric tons per cubic meter, then the material is concrete. If KS *infer_if_mat_is_concrete* determines that the material is concrete, then EVAL asks the user if he/she can provide the parameters and properties of concrete. If any of these parameters and/or properties are unknown, they will be inferred later by EVAL.

Determination of the properties of concrete. If any of the following properties and/or parameters of

concrete is unknown:

- f'c (*fc*),
- fy (*fy*),
- percentage of reinforcing steel (*steel_per*), or
- depth of concrete cover of reinforcing steel (*cover*),

it must be inferred by EVAL. There are four knowledge sources, one for each of the parameters above, that infer the missing information. These four knowledge sources are listed below.

```
*****************************************************************
*                                                               *
* Knowledge Source: infer_fc_of_concrete                        *
*                                                               *
*****************************************************************

Parameter list: year_built -> int
                fc -> real    ==> RESULT
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"fc" is-a --> pressure; units: "lb/in2"

-----------------------------------------------------------------
|year_built >= 1904 AND year_built < 1950             | T . . F|
|year_built >= 1950 AND year_built < 1970             | . T . F|
|year_built >= 1971                                   | . . T F|
|-----------------------------------------------------+--------|
|fc = 3000.0                                          | X      |
|fc = 3500.0                                          |   X    |
|fc = 4000.0                                          |     X  |
|fc = ask the user                                    |       X|
|-----------------------------------------------------+--------|
|explanation = "SEES -> a KBES written by J.F. Peters"| X X X  |
|explanation = "provided by the user at run-time"     |       X|
-----------------------------------------------------------------

*****************************************************************
*                                                               *
* Knowledge Source: infer_fy_of_steel                           *
*                                                               *
*****************************************************************

Parameter list: year_built -> int
                fy -> real    ==> RESULT
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"fy" is-a --> pressure; units: "lb/in2"
```

```
-------------------------------------------------------
|year_built >= 1904 AND year_built < 1970         | T . F|
|year_built >= 1971                               | . T F|
|-------------------------------------------------+------|
|fy = 33000.0                                     | X    |
|fy = 40000.0                                     |   X  |
|fy = ask the user                                |     X|
|-------------------------------------------------+------|
|explanation = "SEES -> a KBES by J.F. Peters"    | X X  |
|explanation = "provided by the user at run-time"|     X|
-------------------------------------------------------


*****************************************************************
*                                                               *
* Knowledge Source: infer_steel_percentage                      *
*                                                               *
*****************************************************************

Parameter list: year_built -> int
                steel_percentage -> real    ==> RESULT
                explanation -> string

---------------------------------------
|year_built < 1950                | T F|
|---------------------------------+----|
|steel_percentage = 1.5           | X  |
|steel_percentage = 1.0           |   X|
|---------------------------------+----|
|explanation = "Inferred by EVAL"| X X|
---------------------------------------

*****************************************************************
*                                                               *
* Knowledge Source: infer_steel_cover_depth                     *
*                                                               *
*****************************************************************

Parameter list: use -> string
                year_built -> int
                depth_of_steel_cover -> real    ==> RESULT
                explanation -> string
                _b_length_units -> string

"depth_of_steel_cover" is-a --> length; units: "in"

-----------------------------------------------------------------
|use = "outdoor"                                  | T T T F F F|
|use = "indoor"                                   | F F F T T T|
|year_built >= 1904 AND year_built < 1920         | T F F T F F|
|year_built >= 1920 AND year_built < 1924         | F T F F T F|
|year_built >= 1924                               | F F T F F F|
|-------------------------------------------------+------------|
|depth_of_steel_cover = 1.0                       | X     X    |
|depth_of_steel_cover = 1.5                       |           X|
|depth_of_steel_cover = 2.0                       |   X X   X  |
|-------------------------------------------------+------------|
|explanation = "SEES -> a KBES written by J. F. Peters"| X X X X X X|
-----------------------------------------------------------------
```

Essentially, the properties and parameters of concrete are derived by the year the material was built. In

a more elaborate highly customized version of EVAL, additional parameters could be easily incorporated to participate in the process of inferring the properties of concrete, including properties such as: geographic location of the building, visible deterioration, and other parameters that refer to additional knowledge for inferring properties of concrete.

Determination of E and G from concrete data. Young's and the shear modulus of concrete can be determined in EVAL from the properties of concrete. The user selects the relationship between the parameters of concrete and the mechanical properties of the material through two KSs: *determine_E_of_concrete* and *determine_G_of_concrete*, listed below:

```
*****************************************************************
*                                                               *
* Knowledge Source: determine_E_of_concrete                     *
*                                                               *
*****************************************************************

Parameter list: is_E_known -> string
                E -> real    ==> RESULT
                fc -> real
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"E" is-a --> pressure; units: "kg/cm2"
"fc" is-a --> pressure; units: "kg/cm2"

-----------------------------------
|is_E_known = "no"                | T|
|---------------------------------+--|
|E = 10000.0 * sqrt(fc)           | X|
|---------------------------------+--|
|explanation = "inferred from f'c"| X|
|is_E_known = "yes"               | X|
-----------------------------------

*****************************************************************
*                                                               *
* Knowledge Source: determine_G_of_concrete                     *
*                                                               *
*****************************************************************

Parameter list: is_G_known -> string
                G -> real    ==> RESULT
                fc -> real
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"G" is-a --> pressure; units: "kg/cm2"
"fc" is-a --> pressure; units: "kg/cm2"

-----------------------------------
|is_G_known = "no"                | T|
|---------------------------------+--|
|G = 2500.0 * sqrt(fc)            | X|
|---------------------------------+--|
|is_G_known = "yes"               | X|
|explanation = "assumed as E/4"   | X|
-----------------------------------
```

The assumptions made for deriving the Young's (E) and shear (G) modulus of concrete are that, according to [4], E can be computed as:

$$E = 10000 \sqrt{f'c}$$

If the shear modulus of a material is unknown, it is assumed in EVAL that its value is a fourth of the Young's modulus

$$G = E / 4.0$$

Determination of the geometric properties of cross sections. If any of the geometric properties of the cross section of a linear element is missing, EVAL computes them according to the formulas contained in the following knowledge source:

```
*****************************************************************
*                                                               *
* Knowledge Source: det_sec_properties                          *
*                                                               *
*****************************************************************

Parameter list: b -> real
                d -> real
                radius -> real
                is_area_known -> string
                is_Iy_known -> string
                is_Iz_known -> string
                is_J_known -> string
                section_type -> string
                area -> real
                Iy -> real
                Iz -> real
                J -> real
                area_source -> string
                Iy_source -> string
                Iz_source -> string
                J_source -> string
                _b_length_units -> string

"b" is-a --> length; units: "in"
"d" is-a --> length; units: "in"
"area" is-a --> area; units: "in2"
"Iy" is-a --> inertia; units: "in4"
"Iz" is-a --> inertia; units: "in4"
"J" is-a --> inertia; units: "in4"
```

```
-------------------------------------------------
|is_area_known = "no"              | T . . . T . . .|
|is_Iy_known = "no"                | . T . . . T . .|
|is_Iz_known = "no"                | . . T . . . T .|
|is_J_known = "no"                 | . . . T . . . T|
|section_type = "rectangular"      | T T T T . . . .|
|section_type = "circular"         | . . . . T T T T|
|----------------------------------+----------------|
|area = b * d                      | X              |
|Iy = d*pow(b,3.0)/12.0            |   X            |
|Iz = b*pow(d,3.0)/12.0            |     X          |
|J = J_of_rect_secs(b,d)           |       X        |
|area = 3.141593 * radius * radius|         X      |
|Iy = 3.141593/4.0*pow(radius,4.0)|           X    |
|Iz = 3.141593/4.0*pow(radius,4.0)|             X  |
|J = 3.141593/2.0*pow(radius,4.0)  |               X|
|is_area_known = "yes"             | X       X      |
|is_Iy_known = "yes"               |   X       X    |
|is_Iz_known = "yes"               |     X       X  |
|is_J_known = "yes"                |       X       X|
|area_source = "computed by EVAL"  | X       X      |
|Iy_source = "computed by EVAL"    |   X       X    |
|Iz_source = "computed by EVAL"    |     X       X  |
|J_source = "computed by EVAL"     |       X       X|
-------------------------------------------------
```

```
Control strategy for firing rules --> fire all.
```

The KS *det_sec_parameters* works only with rectangular and circular sections. Other types of sections could be easily incorporated by defining a KS with the proper formulas for the new sections. In the current implementation of EVAL, the geometric properties of unknown section types are asked directly to the user.

6.2 Evaluation of Horizontal Subsystems

There are two sub-tasks in the process of evaluating the horizontal subsystems, illustrated in Figure 5-19, that are subject to customization: the classification of the material of the slabs and the evaluation of the connection between two slabs. This tasks are related to several KSs, as shown in Figure 6-2.

Classification of the materials in the slabs. The materials classified as rigid will be assumed to be infinitely rigid; the materials classified as not rigid will be assumed to have no in-plane stiffness. This classification is made using the name and the density of the material. The knowledge source that classifies the materials is *det_mat_rigidity*:

```
*****************************************************************
*                                                               *
* Knowledge Source: det_mat_rigidity                            *
*                                                               *
*****************************************************************

Parameter list: material_name -> string
                density -> real
                is_material_concrete -> string
                rigid -> string ==> RESULT
                explanation -> string
```

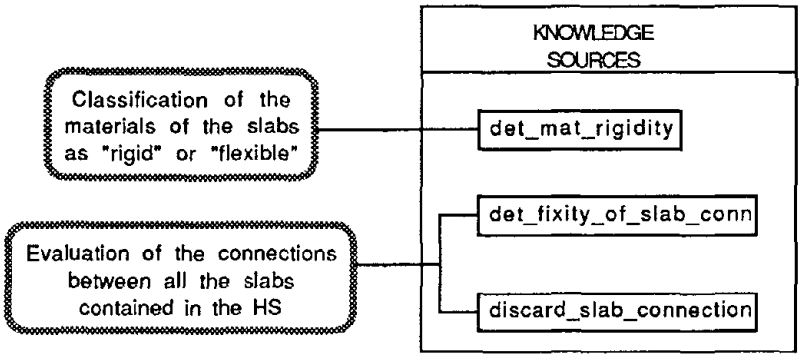


Figure 6-2: Customization of the process of evaluating the horizontal subsystems

```
                _b_force_units -> string
                _b_length_units -> string

If the user is requested to provide a value for the variable
"rigid", the value provided by the user will be limited to:

    1) yes                              2) no


The parameter "is_material_concrete" is derived by KS: "infer_if_mat_is_concrete
                material_name -> sent as "material_name"
                density -> sent as "density"
                is_material_concrete -> sent as "is_concrete"
                explanation -> sent as "explanation"
                _b_force_units -> sent as "_b_force_units"
                _b_length_units -> sent as "_b_length_units"

"density" is-a --> density; units: "Ton/m3"

-------------------------------------------------------------------
|is_material_concrete = "yes"                    | . T F F| ELSE |
|density > 1.5                                   | F T . .|      |
|material_name = "steel-deck"                    | . F T F|      |
|material_name = "wood"                          | . F F T|      |
|------------------------------------------------+--------+------|
|rigid = "yes"                                   |   X    |      |
|rigid = "no"                                    | X   X X|      |
|rigid = ask the user                            |        |   X  |
|------------------------------------------------+--------+------|
|explanation = "inferred by EVAL"                | X X X X|      |
|explanation = "provided by the user at run-time"|        |   X  |
-------------------------------------------------------------------
```

KS *det_mat_rigidity* contains a very small list of materials. This list is mainly illustrative and a larger and more complete list can be easily incorporated. The process of classifying the materials of the slabs can easily involve additional issues if the knowledge is provided.

Evaluation of the connections between slabs. The purpose of this sub-task is to approve or

disqualify the connection between two individual slabs. If any connection between two slabs within a horizontal subsystem is disapproved, the slabs are regrouped again considering only the approved connections; this may lead to the decomposition of the current horizontal subsystem into new smaller horizontal subsystems. There are two knowledge sources that control the process of evaluating the connections: *det_fixity_of_slab_conn* and *discard_slab_connection*.

```
*****************************************************************
*                                                               *
* Knowledge Source: det_fixity_of_slab_conn                     *
*                                                               *
*****************************************************************

Parameter list: mat1_rigid -> string
                mat2_rigid -> string
                thick1 -> real
                thick2 -> real
                mat_slab1 -> string
                mat_slab2 -> string
                fixed -> string ==> RESULT
                explanation -> string

-----------------------------------------------------------------
|mat1_rigid != mat2_rigid                          | F T F| ELSE |
|rmin(thick1,thick2)/rmax(thick1,thick2) < 0.75    | F . T|      |
|--------------------------------------------------+------+------|
|fixed = "yes"                                     | X    |      |
|fixed = "no"                                      |   X X|      |
|fixed = ask the user                              |      |   X  |
|--------------------------------------------------+------+------|
|explanation = "inferred by EVAL"                  | X X X|      |
|explanation = "provided by the user at run-time"  |      |   X  |
-----------------------------------------------------------------

*****************************************************************
*                                                               *
* Knowledge Source: discard_slab_connection                     *
*                                                               *
*****************************************************************

Parameter list: fixed -> string
                mat_slab1 -> string
                mat_slab2 -> string
                mat1_rigid -> string
                mat2_rigid -> string
                approve_slab_conn -> string ==> RESULT
                explanation -> string

-------------------------------------------
|fixed = "no"                   | T .| ELSE |
|mat1_rigid != mat2_rigid       | . T|      |
|-------------------------------+----+------|
|approve_slab_conn = "yes"      |    |   X  |
|approve_slab_conn = "no"       | X X|      |
|-------------------------------+----+------|
|explanation = "inferred by EVAL"| X X|      |
-------------------------------------------
```

The first KS, *det_fixity_of_slab_conn* defines whether a connection can be considered fixed, that is, the two connected slabs act as a unit. The second KS, *discard_slab_connection*, decides whether a

connection is approved or disqualified.

6.3 Determination of Redundancy in the Transmission of Seismic Loads

This section presents the definition of redundancy in the tranfer of seismic loads implemented in the present version of EVAL. The existence of redundancy in the transfer of seismic loads is verified for each orthogonal direction (X,Y) separately. As it can be seen in Figure 5-34, the parameters provided to the user for defining redundancy are the following:

- the number of elements supporting the horizontal subsystem in the current direction; and
- the maximum percentage of the seismic load acting along the current direction carried by an individual element.

The definition of redundancy is expressed in the knowledge source *def_HS_redundancy*, presented below.

```
*************************************************************
*                                                           *
* Knowledge Source: def_HS_redundancy                       *
*                                                           *
*************************************************************

Parameter list: n_elements_resisting_F -> int
                MAX_percentage_of_F_in_el -> real
                HS_has_redundancy -> string ==> RESULT
                explanation -> string
```

n_elements_resisting_F < 3	T	.	ELSE
MAX_percentage_of_F_in_el > 50.0	.	T	
HS_has_redundancy = "yes"			X
HS_has_redundancy = "no"	X	X	
explanation = "less than 3 elements resisting the seismic force"	X		
explanation = "a single element resists more than 50% of the sei		X	

The default knowledge source *def_HS_redundancy* in EVAL states that if a horizontal subsystem has less than three elements supporting it, or if a single element carries more than the 50% of the seismic load acting on the horizontal subsystem, then there is no redundancy in the transfer of the seismic load acting on the current horizontal subsystem.

6.4 Evaluation of Torsion in Horizontal Subsystems

As it can be seen in Figure 5-36, in the process of evaluating torsion in a horizontal subsystem, the user has control over two issues:

- defining when a horizontal subsystem has a significant torsion problem; and
- the listing of elements with a significant contribution to the torsional stability of a horizontal subsystem.

The parameters provided for diagnosing a relevant torsion problem in a building are:

- the location of the center of weight of the horizontal subsystem (Xm,Ym);
- the location of the center of torsion of the horizontal subsystem (Xt,Yt); and
- the dimensions of the horizontal subsystem along the X and Y direction (XL and YL).

Knowledge source *identify_torsion_problem*, presented below, defines whether a horizontal subsystem has a significant torsion problem or not.

```
************************************************************
*                                                          *
* Knowledge Source: identify_torsion_problem               *
*                                                          *
************************************************************

Parameter list: XL -> real
                YL -> real
                Xm -> real
                Ym -> real
                Xt -> real
                Yt -> real
                torsion_problem -> string ==> RESULT
                explanation -> string
```

rabs(Xm-Xt) >= 0.2*XL OR rabs(Ym-Yt) >= 0.2*YL	T	ELSE
torsion_problem = "yes"	X	
torsion_problem = "no"		X
explanation = "excessive distance between the centers of weight an	X	

Basically, the criteria in the above knowledge source states that if the distance, in any direction, between the centers of weight and torsion of the horizontal subsystem is greater than 20% of the corresponding dimension, the horizontal subsystem has a significant torsion problem.

For reporting the elements that make the most significant contribution to the torsional stability of a horizontal subsystem previously diagnosed with a significant torsion problem, the following knowledge sources is used.

```
************************************************************
*                                                          *
* Knowledge Source: rep_torsion_element                    *
*                                                          *
************************************************************

Parameter list: HS_is_flexible -> string
                torsion_percentage -> real
                report_element -> string ==> RESULT
                explanation -> string
```

```
-----------------------------------------
|HS_is_flexible = "yes"          | T F| ELSE |
|rabs(torsion_percentage) >= 25.0| T .|      |
|rabs(torsion_percentage) >= 50.0| . T|      |
|--------------------------------+----+------|
|report_element = "yes"          | X X|      |
|report_element = "no"           |    |   X  |
-----------------------------------------
```

The elements with the most significant contribution to the stability of a horizontal subsystem are included in the evaluation report. The user controls with the above knowledge source the threshold for determining the elements to be included in the evaluation report.

6.5 Evaluation of the Resistance of Individual Elements and Connections

This section presents the customizable procedures of determining whether local and global force components acting on structural elements must be released. The contents of the knowledge sources in this section are mainly illustrative and correspond to an initial approximation of more precise and detailed procedures.

Releases in local coordinate system. It can be seen in Figure 5-41 that a knowledge source is used for each of the four type of possible forces acting on a structural element. The following is the meaning of the relevant parameters used in the knowledge sources that release components in local coordinate system.

- *Mx:* torsional moment acting on the element;
- *Mf:* flexural moment acting on the element;
- *force_magnitude:* magnitude of a shear or axial force acting on the element;
- *b, h:* dimensions of the beam;
- *d:* depth of the concrete cover of reinforcing steel; the distance from the extreme of the cross section subject to compression to the reinforcing steel is equal to h-d;
- *P:* percentage of reinforcing steel in the cross section;
- *fc, fy: f'c* of concrete and *fy* of reinforcing steel;
- *R:* radius of the section, used only if the section is circular;
- *E:* E is the Young's modulus of the material;
- *I:* moment of inertia of the section in the direction of the moment;
- *area:* area of the cross section; and
- *release:* main result of the knowledge source (*yes* or *no*).

The knowledge source *release_axial_component*, listed below, determines whether the axial force

acting on the end of an element should be released or not.

```
*****************************************************************
*                                                               *
* Knowledge Source: release_axial_component                     *
*                                                               *
*****************************************************************

Parameter list: material_is_concrete -> string
                force_magnitude -> real
                area -> real
                fc -> real
                release_axial -> string ==> RESULT
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"force_magnitude" is-a --> force; units: "kg"
"area" is-a --> area; units: "cm2"
"fc" is-a --> pressure; units: "kg/cm2"

-------------------------------------------------
|material_is_concrete = "yes"         | T F| ELSE |
|rabs(force_magnitude/area) >= 0.6*fc| T .|      |
|rabs(force_magnitude/area) >= 120.0 | . T|      |
|------------------------------------+----+------|
|release_axial = "yes"                | X X|      |
|release_axial = "no"                 |    |   X  |
-------------------------------------------------
```

The criteria in knowledge source *release_axial_component* is arbitrary; the allowable axial stress was set to 60% of f'c, if the material is concrete, or to 120 [kg/cm2], for any other type of material. Other types of materials or more accurate procedures for determining axial capacity could be included.

The shear forces are evaluated by knowledge source *release_shear_component*. The criteria for releasing shear force components is presented below.

```
*****************************************************************
*                                                               *
* Knowledge Source: release_shear_component                     *
*                                                               *
*****************************************************************

Parameter list: material_is_concrete -> string
                force_magnitude -> real
                area -> real
                fc -> real
                release_shear -> string ==> RESULT
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"force_magnitude" is-a --> force; units: "kg"
"area" is-a --> area; units: "cm2"
"fc" is-a --> pressure; units: "kg/cm2"
```

```
-------------------------------------------------------
|material_is_concrete = "yes"                | T F| ELSE |
|rabs(force_magnitude)/area >= 0.5*sqrt(fc)| T .|      |
|rabs(force_magnitude)/area >= 7.5           | . T|      |
|--------------------------------------------+----+------|
|release_shear = "yes"                       | X X|      |
|release_shear = "no"                        |    |   X  |
-------------------------------------------------------
```

The allowable shear stress was set to half the square root of the f'c if the material is concrete, or to 7.5 [kg/cm2] if the material is not concrete. The following formula for estimating the allowable shear stress in concrete elements:

$$vc = 0.5\sqrt{f'c}$$

was taken from [20] and corresponds to an approximation of the shear resistance of a concrete section without shear reinforcement. The value of 7.5 [kg/cm2] was selected arbitrarily.

The torsional moment is evaluated by knowledge source *release_torsion_component*, presented below.

```
*****************************************************************
*                                                               *
* Knowledge Source: release_torsion_component                   *
*                                                               *
*****************************************************************

Parameter list: material_is_concrete -> string
                section_type -> string
                Mx -> real
                b -> real
                h -> real
                radius -> real
                fc -> real
                area -> real
                J -> real
                release_torsion_component -> string ==> RESULT
                explanation -> string
                _b_force_units -> string
                _b_length_units -> string

"Mx" is-a --> moment; units: "kg*cm"
"b" is-a --> length; units: "cm"
"h" is-a --> length; units: "cm"
"radius" is-a --> length; units: "cm"
"fc" is-a --> pressure; units: "kg/cm2"
"area" is-a --> area; units: "cm2"
"J" is-a --> inertia; units: "cm4"
```

```
-------------------------------------------------------------------------
|material_is_concrete = "yes"                                | T T T T T T F F|
|section_type = "rectangular"                                | T T F F F F . .|
|section_type = "circular"                                   | F F T T F F . .|
|rabs(Mx) >= 0.6*rmin(b,h)*rmin(b,h)*rmax(b,h)*sqrt(fc)      | T F . . . . . .|
|rabs(Mx) >= 0.1*3.1416*pow(radius*2.,3.)*sqrt(fc)           | . . T F . . . .|
|rabs(Mx)/J*sqrt(area/3.1416) >= 0.6*fc                      | . . . . T F . .|
|rabs(Mx)/J*sqrt(area/3.1416) >= 120.0                       | . . . . . . T F|
|------------------------------------------------------------+----------------|
|release_torsion_component = "yes"                          | X   X   X   X  |
|release_torsion_component = "no"                           |   X   X   X   X|
-------------------------------------------------------------------------
```

The criteria in knowledge source *release_torsion_component* contain the following cases:

- rectangular sections of concrete;
- circular sections of concrete;
- cross sections, other than rectangular or circular, made of concrete;
- sections not made of concrete.

For rectangular sections of concrete, the following formula, taken from [20], is used to compute the allowable torsion moment in a given element.

$$Mx <= 0.6\ b^2\ h\ \sqrt{f'c}$$

In the above formula, Mx is the actual torsional moment acting on the element. In knowledge source *release_torsion_component*, the parameter b in the above formula is taken as MIN(b,h), and h as MAX(b,h), to be in the conservative side. For circular sections of concrete, the following expression is used to compute the allowable torsion moment.

$$Mx <= 0.1\ \pi\ d^3\ \sqrt{f'c}$$

The above formula was taken from [20] and, as the formula for rectangular sections, is based in the theory developed by Hsu [25] about failure mechanisms in concrete elements subject to pure torsion. For concrete sections other than circular or rectangular and for sections not made of concrete, the allowable torsional stress is set to be the same as the shear stress defined in *release_shear_component*.

Knowledge source *release_flexural_moment* is responsible for releasing the force components that correspond to flexural moment. These components, in local coordinate system, are My' and Mz'. Knowledge source *release_bending_moment* is presented below.

```
************************************************************
*                                                          *
* Knowledge Source: release_bending_moment                 *
*                                                          *
************************************************************

Parameter list: material_is_concrete -> string
                section_type -> string
                element_type -> string
```

```
                Mf -> real
                b -> real
                h -> real
                d -> real
                P -> real
                fc -> real
                fy -> real
                R -> real
                E -> real
                I -> real
                area -> real
                release -> string ==> RESULT
                _b_force_units -> string
                _b_length_units -> string

"Mf" is-a --> moment; units: "kg*cm"
"b" is-a --> length; units: "cm"
"h" is-a --> length; units: "cm"
"d" is-a --> length; units: "cm"
"fc" is-a --> pressure; units: "kg/cm2"
"fy" is-a --> pressure; units: "kg/cm2"
"R" is-a --> length; units: "cm"
"E" is-a --> pressure; units: "kg/cm2"
"I" is-a --> inertia; units: "cm4"
"area" is-a --> area; units: "cm2"

-------------------------------------------------------------------------
|material_is_concrete = "yes"                 | T T T T T T F F F F . .|
|section_type = "rectangular"                 | T T T T F F T T F F F F|
|section_type = "circular"                    | F F F F T T F F T T F F|
|element_type = "beam"                        | T T F F . . . . . . . .|
|rabs(Mf) >= Mr_in_minor_axis(fc,fy,b,h,d,P)  | T F . . . . . . . . . .|
|rabs(Mf) >= Mr_of_rect_col(fc,fy,b,h,d,P)    | . . T F . . . . . . . .|
|rabs(Mf) >= Mr_of_circular_beam(fc,fy,R,d,P)| . . . . T F . . . . . .|
|rabs(Mf) >= 0.0009*E*I/h                     | . . . . . . T F . . . .|
|rabs(Mf) >= 0.00045*E*I/R                    | . . . . . . . . T F . .|
|rabs(Mf) >= 0.00045*E*I/sqrt(area)           | . . . . . . . . . . T F|
|---------------------------------------------+----------------------|
|release = "yes"                              | X   X   X   X   X   X  |
|release = "no"                               |   X   X   X   X   X   X|
-------------------------------------------------------------------------
```

As in the case of the release of torsion, several cases are included in the knowledge source *release_bending_moment*:

- continuously reinforced concrete columns with rectangular sections;
- simply reinforced concrete beams with rectangular sections and bending with respect the minor axis;
- circular sections of reinforced concrete;
- rectangular sections not made of concrete;
- *circular sections not made of concrete;*
- sections other than rectangular or circular, regardless of the type of material that these sections are made of.

The reinforcement in columns and diagonals made of reinforced concrete and with rectangular sections is assumed to be uniformly distributed around the rectangular section, at a distance *d* from the external perimeter of the section. The depth of the uniform reinforcement, *t*, is computed. Beam elements are subject to bending moment around their minor axis. Beams are assumed to be singly reinforced. The assumptions for columns and bending of beams with respect their minor axis is illustrated in Figure 6-3. The resistance of sections of the type shown in Figure 6-3 is computed using the iterative balanced section method.

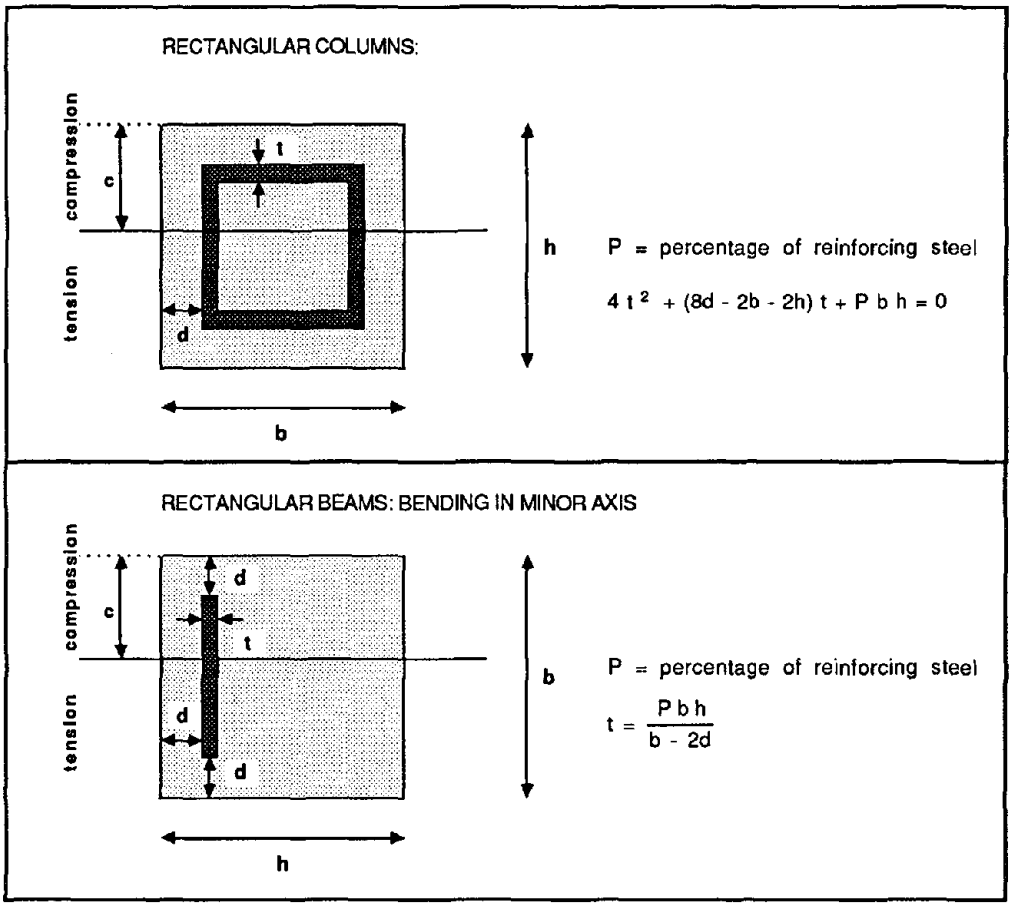


Figure 6-3: Assumptions made for computing the flexural resistance, using the balanced section method, of rectangular sections of reinforced concrete

For circular sections, a procedure was implemented to compute the allowable flexural moment using the iterative method of the balanced section [20]. Since the exact location of the reinforcement bars in the section is not known, it is assumed that the reinforcement steel is distributed uniformly around the circular section, at a distance *d* from the external perimeter of the section. The depth of the assumed steel ring, *t*, is computed. Figure 6-4 illustrates the assumption made for computing the allowable flexural moment in circular sections of reinforced concrete. The method of the balanced section consists of finding iteratively

the depth of the neutral axis of the section; the neutral axis is found when the tension and compression resistances are the same for a given depth of the neutral axis. Since the steel is assumed to be distributed uniformly around the element, there is a portion of the steel ring that contributes to tension and another portion that contributes to compression. This method was implemented as an external C function made available to the knowledge acquisition facility of EVAL; the function that computes the allowable flexural moments of circular elements of reinforced concrete is called *Mr_of_circular_beams*.

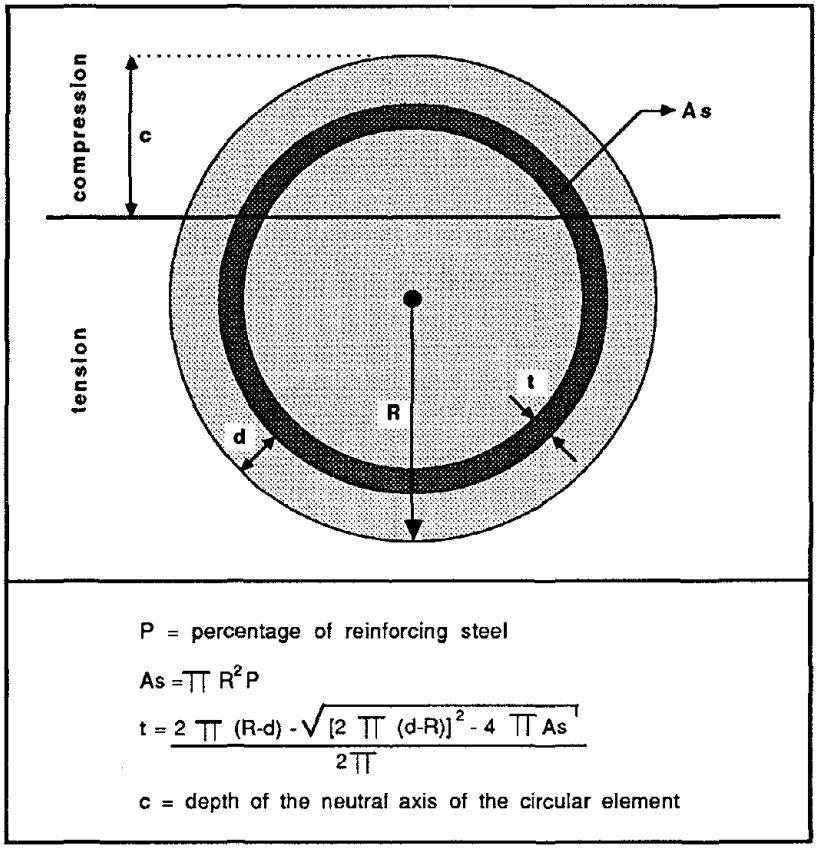


Figure 6-4: Assumptions made for computing the flexural resistance, using the balanced section method, of a circular section of reinforced concrete

For rectangular sections not made of concrete, the assumptions illustrated in Figure 6-5 are made to compute the maximum allowable flexural moment in a given section.

A similar assumption as the one shown in Figure 6-5 is made to compute the maximum allowable flexural moment in circular and arbitrary sections not made of concrete. The formulas for the latter two cases are listed below.

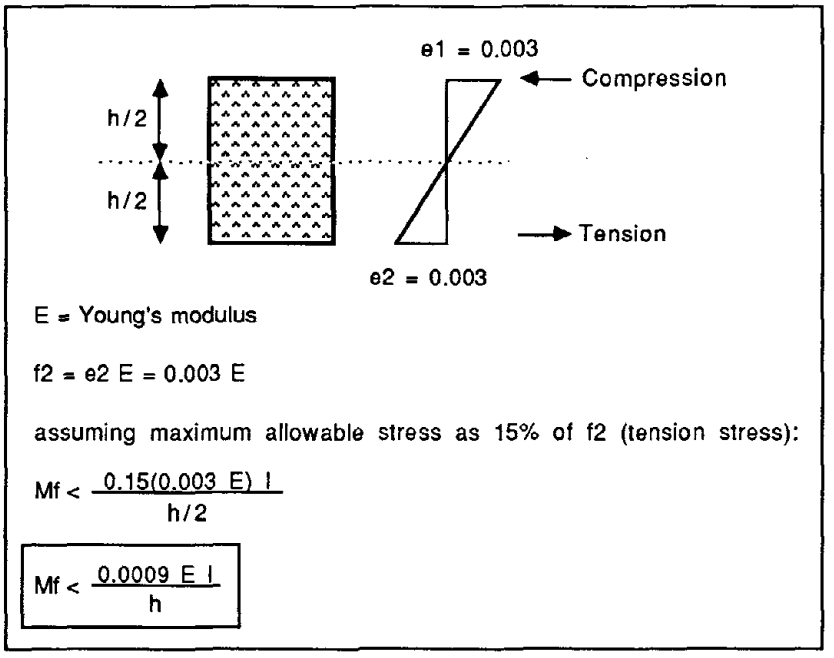


Figure 6-5: Assumptions made for computing the maximum flexural moment in a rectangular section made of a material different than concrete

Circular section: $Mf < \frac{0.00045\ E\ I}{R}$
Non-circular and non-rectangular section: $Mf < \frac{0.00090\ E\ I}{\sqrt{area}}$

Releases in global coordinate system. As it is illustrated in Figure 5-40, a global release is determined by:

- rotating the force components from global to local coordinate system;
- determining the releases of components in local coordinate system;
- rotating the force components in local coordinate system, including the releases, back to global coordinate system; and
- comparing the original force components in global coordinate system, to the components obtained from rotating the released components from local to global coordinate system.

Knowledge source *set_per_for_release*, presented below, compares the values of a force component in global coordinate system before and after the local releases are made.

```
*****************************************************************
*                                                               *
* Knowledge Source: set_per_for_release                         *
*                                                               *
*****************************************************************

Parameter list: force_before_rel -> real
                force_after_rel -> real
                release -> string ==> RESULT
```

rabs(force_before_rel-force_after_rel)/force_before_rel > 0.35	T	ELSE
release = "yes"	X	
release = "no"		X

The criteria in *set_per_for_release* says that a force component in global coordinate system is released if after including the releases made in local coordinate system, the value of that component is changed more than 35% of its original value.

SECTION 7
EXAMPLES OF EVAL EXECUTION

This section illustrates the evaluation results produced by EVAL in several sample runs.

7.1 Problem Description

The hypothetical building used in this section is illustrated in Figures 7-1 and 7-2. The output created by EVAL for the evaluations performed, as well as the input file that describes the building are presented in Chapter 5 of [27].

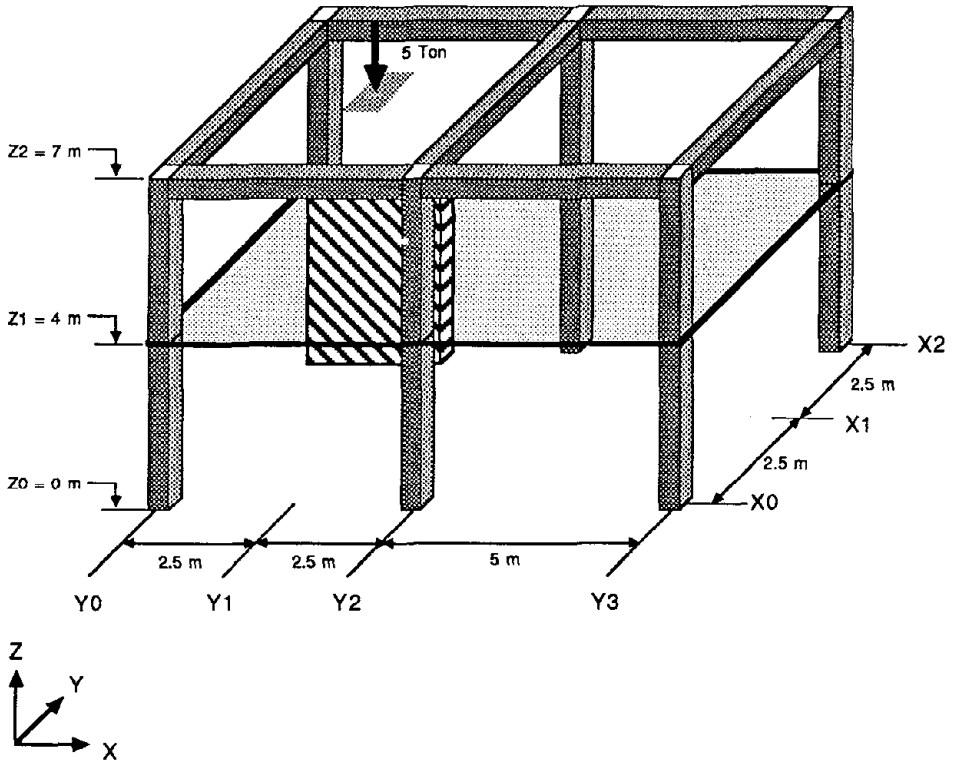


Figure 7-1: Sample building

7.2 First Evaluation Example

The first example consists of executing all the possible evaluation processes on the sample building assuming that the slabs in both floors are made of a rigid material. According to the custom knowledge in EVAL (KS *det_mat_rigidity*), slabs made of reinforced concrete are assumed to be rigid, so it will be specified that both slabs are made of concrete with a density of 2.4 [Ton/M^3]. The beams, columns and the wall in the building are assumed to be made of the same concrete material. The known data of this

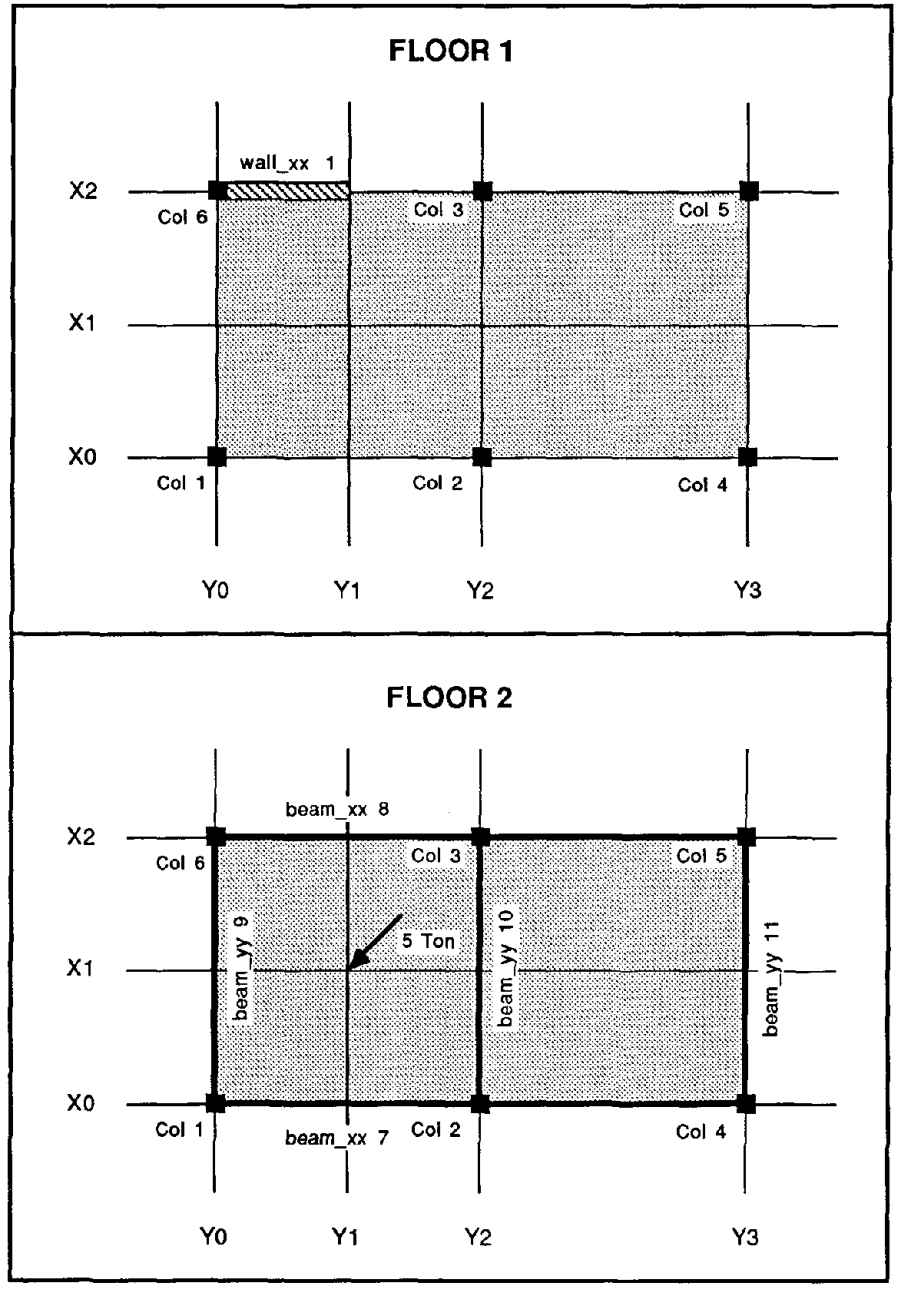


Figure 7-2: Floor plans of the sample building

example provided to EVAL are shown in Figure 7-3. The input file that describes the example of this section is listed in the first section of Chapter 5 of [27]. The context generated by the pre-processor of EVAL from the information in the input file of this example is presented in Chapter 5 of [27].

Determination of material and cross section properties. The data provided to EVAL about the

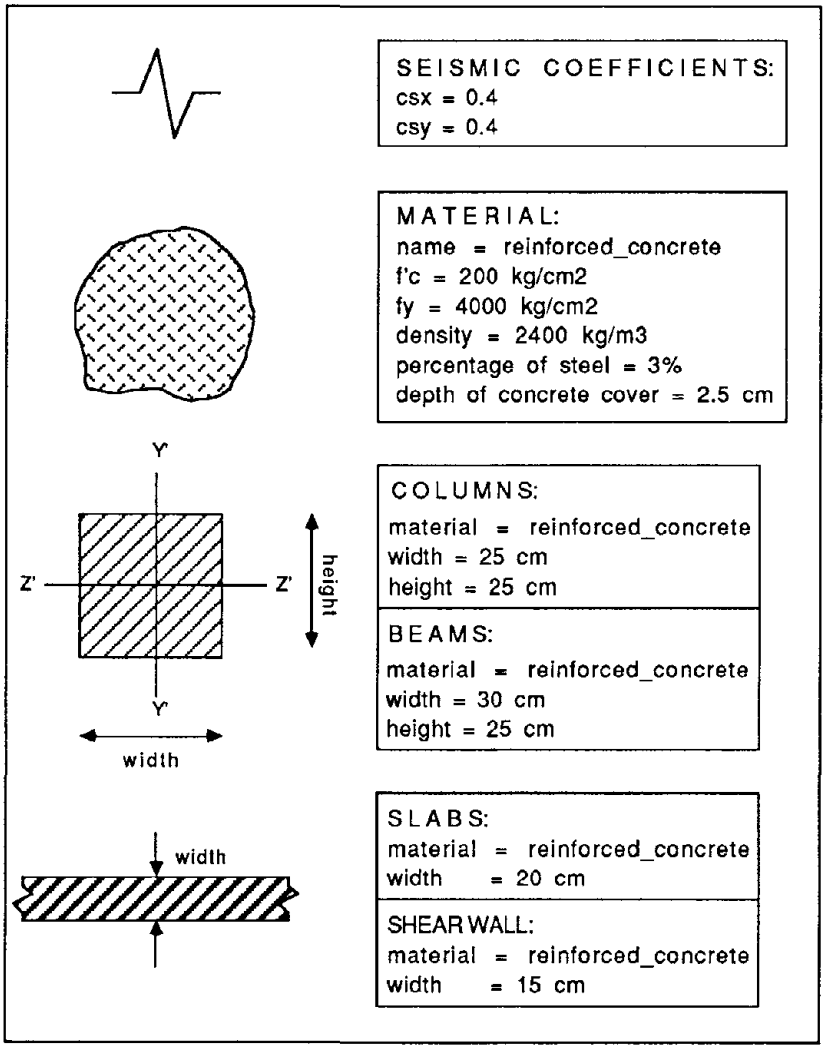


Figure 7-3: Data of the first evaluation case

material and cross sections is illustrated in Figure 7-3. The Young's (E) and shear (G) moduli as well as the geometric properties of the cross sections were not provided. This information was inferred by EVAL and the results were the following:

```
MATERIAL: reinforced_concrete
        material type -> concrete
        E = 1414213.56 Ton/M²
        G =  353553.39 Ton/M²

LINEAR SECTION: beams
        area = 0.075000 M²
        Iy   = 0.000563 M⁴
        Iz   = 0.000391 M⁴
        J    = 0.000775 M⁴
```

```
LINEAR SECTION: columns
        area = 0.062500 M²
        Iy   = 0.000326 M⁴
        Iz   = 0.000326 M⁴
        J    = 0.000550 M⁴
```

Quick evaluation. The quick evaluation of the building produces the following results:

```
Weight of the building =          53.00 [TON]

Overturning moment in the X direction =        67.46 [TON-M]
Overturning moment in the Y direction =        67.46 [TON-M]

Center of gravity of the building: Xm =         4.76 [M]
                                   Ym =         2.50 [M]

Base shears: VX =          10.96 [TON]
             VY =          10.96 [TON]
```

The weight of the building is 53 metric tons, including the superimposed load in the second floor. The center of gravity is shifted slightly to the left of the geometric center of the building due to the superimposed load in the second floor. The seismic coefficients of 0.4 in both directions cause base shears of 10.96 Tons; the seismic forces cause overturning moments in both directions of 67.46 Ton-M.

Evaluation of redundancy in horizontal subsystems. There are two horizontal subsystems in the building. Each horizontal subsystem is identified by an integer number. The horizontal subsystem labeled with number 1 corresponds to the first floor; horizontal subsystem number two corresponds to the second floor of the building.

The total seismic shear acting just below horizontal subsystem 1 has a magnitude of 10.96 Tons. The summary of the redundancy evaluation of horizontal subsystem 1 is presented in the table of Figure 7-4.

Element Type and ID	Force in X Direction	Contribution in X direction	Force in Y Direction	Contribution in Y direction
Column 1	0.02 Ton	0.17 %	1.83 Ton	16.68 %
Column 2	0.02 Ton	0.17 %	1.83 Ton	16.68 %
Column 3	0.02 Ton	0.17 %	1.83 Ton	16.68 %
Column 4	0.02 Ton	0.17 %	1.83 Ton	16.68 %
Column 5	0.02 Ton	0.17 %	1.83 Ton	16.68 %
Wall_xx 1	10.86 Ton	99.15 %	1.82 Ton	16.60 %
SUM	10.96 Ton	100.00 %	10.96 Ton	100.00 %

Figure 7-4: Summary of the evaluation of redundancy in horizontal subsystem 1

The diagnostic produced by EVAL for horizontal subsystem 1 is:

- for the X direction: redundancy evaluation was not satisfactory; and
- for the Y direction: redundancy evaluation was satisfactory.

It can be seen in the table of Figure 7-4 that the shear wall takes almost the entire seismic shear in the X direction (99.15 %). This fact causes the redundancy check to fail in the X direction. According to KS *def_HS_redundancy*, a horizontal subsystem lacks sufficient redundancy in the transfer of shear loads if a single element supporting the horizontal subsystem carries more than 50% of the total seismic shear in the direction of interest. In the Y direction, the distribution of the seismic shear among the elements that support horizontal subsystem 1 is even and the redundancy evaluation in the Y direction is satisfactory.

The seismic forces acting on horizontal subsystem 2 in both directions (X, Y) have 7.88 Tons of magnitude. Since horizontal subsystem 2 corresponds to the top floor of the building, the seismic forces and shears have the same magnitude. The evaluation of redundancy in horizontal subsystem 2 was satisfactory for both directions. The summary of the redundancy evaluation of horizontal subsystem 2 is presented in the table of Figure 7-5.

Element Type and ID	Force in X Direction	Contribution in X direction	Force in Y Direction	Contribution in Y direction
Column 1	1.31 Ton	16.67 %	1.31 Ton	16.67 %
Column 2	1.31 Ton	16.67 %	1.31 Ton	16.67 %
Column 3	1.31 Ton	16.67 %	1.31 Ton	16.67 %
Column 4	1.31 Ton	16.67 %	1.31 Ton	16.67 %
Column 5	1.31 Ton	16.67 %	1.31 Ton	16.67 %
Column 6	1.31 Ton	16.67 %	1.31 Ton	16.67 %
SUM	7.86 Ton	100.00 %	7.86 Ton	100.00 %

Figure 7-5: Summary of the evaluation of redundancy in horizontal subsystem 2

It can be seen in the redundancy evaluation of the horizontal subsystems that the distribution of seismic shears among the elements that support a horizontal subsystem is made according to the stiffness of each element. Since the 2D analysis model is used for this analysis, the position of an element within the floor plan does not affects its percentage of contribution since torsion is ignored.

Torsion check. Horizontal subsystem 1 was diagnosed to have a significant torsion problem. The positions of the centers of gravity and torsion of horizontal subsystem 1 were found to be the following:

```
Horizontal subsystem 1 has a significant torsion problem:

     Center of gravity 1 -> Xm =      5.00 [M]   Ym =      2.50 [M]
     Center of torsion 1 -> Xt =      6.24 [M]   Yt =      4.11 [M]
```

The position of the center of torsion in horizontal subsystem 1 has an offset, in the Y direction, of 32.2% of the Y dimension of the horizontal subsystem from the center of gravity. According to KS *identify_torsion_problem*, if the distance between the center of gravity and the center of torsion of a horizontal subsystem exceeds 20% of the corresponding direction, the horizontal subsystem is subject to excessive torsion. The element making the most significant contribution to the torsional stability of horizontal subsystem 1 is the shear wall, carrying -86.71% of the torsional moment acting on the

horizontal subsystem. The meaning of the percentage of contribution to the torsional stability is explained in section 5.

Horizontal subsystem 2 passed the torsion evaluation. This means that the position of the center of torsion of horizontal subsystem 2 is within the limits set by KS *identify_torsion_problem* with respect the center of gravity. The elements making the most significant contribution to the torsional stability of horizontal subsystem 2 are columns 4 and 6, contributing respectively with 58.82% and -61.31% of the torsional moment acting on the horizontal subsystem.

Reactions at the supports. The reactions at the supports are summarized in Figure 7-6. These reactions are computed according to the 3D model of analysis.

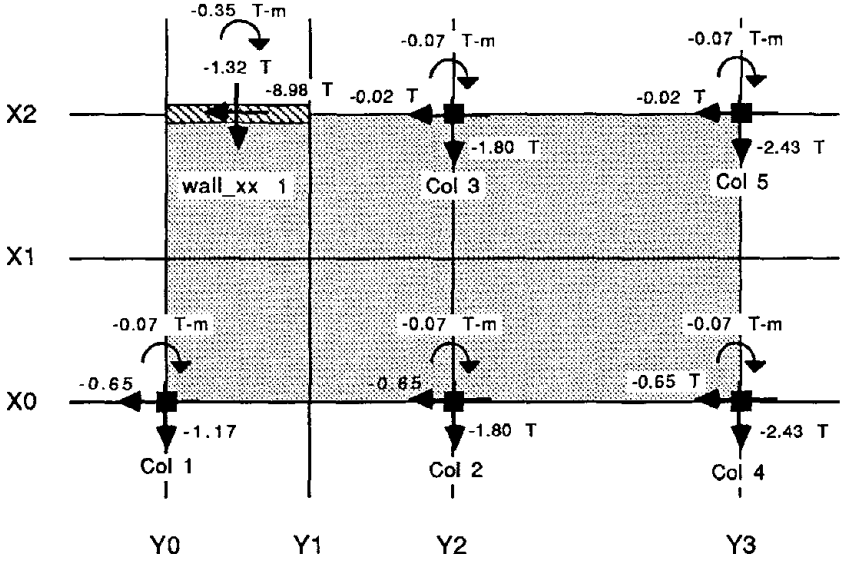


Figure 7-6: Reactions at the supports in the first evaluation example

Evaluation of the resistance of individual elements and connections. In the current example, none of the internal forces on individual members exceeded the capacity of the member, so no force components were released. The effect of releases will be illustrated in the next example.

7.3 Second Evaluation Example

In this section, the building used in the prior section of this section will be evaluated for torsion and redundancy but specifying the release of force components prior to the evaluation. In this example, the transmission of force in the Y direction will be disabled in the second floor for columns 4 and 5. The releases are interpreted as follows:

- column 4 cannot transmit force in the Y direction at point (X0,Y3,Z2), just below the slab in the second floor; and
- column 5 cannot transmit force in the Y direction at point (X2,Y3,Z2), just below the slab in the second floor.

The following is the information reported by EVAL about the releases described above:

```
RELEASE MADE IN: column 4

   released in segment from point X0 Y3 Z1 to X0 Y3 Z2
   in point X0 Y3 Z2
            transmission of force in the Y direction released

RELEASE MADE IN: column 5

   released in segment from point X2 Y3 Z1 to X2 Y3 Z2
   in point X2 Y3 Z2
            transmission of force in the Y direction released
```

The above releases originate a torsion problem in the second floor. The results of the evaluation of torsion in horizontal subsystem 2 are summarized below:

```
Horizontal subsystem 2 has a significant torsion problem.

Center of gravity 2 -> Xm =       4.57 [M]    Ym =       2.50 [M]
Center of torsion 2 -> Xt =       3.30 [M]    Yt =       1.30 [M]
```

With columns 4 and 5 in the second floor released, columns 2 and 6 make the most significant contributions to the torsional stability of horizontal subsystem 2 (column 2 contributes with +55.09% and column 6 with -70.19%). The redundancy and torsion evaluations of this example are shown in Chapter 5 of [27].

7.4 Third Evaluation Example

In this section, the following modifications were made to the original data of the building:

- the value of f'c of concrete was assumed to be 120 [kg/cm^2] implying an extremely weak or highly corroded structure; and
- the cross section of column 4 was changed to a rectangular section of 15x15 cm to introduce the effect of unequal distribution of the supporting elements.

With the above modifications, the following releases of force components were made successively by EVAL:

```
RELEASES IN: column 1

   released in segment from point X0 Y0 Z0 to X0 Y0 Z1
   in point X0 Y0 Z1
            transmission of force in X direction released
            transmission of force in Y direction released
            transmission of moment around Z axis released
```

```
RELEASES IN: column 2

   released in segment from point X0 Y2 Z0 to X0 Y2 Z1
   in point X0 Y2 Z1
            transmission of force in X direction released
            transmission of force in Y direction released
            transmission of moment around Z axis released


RELEASES IN: column 3

   released in segment from point X2 Y2 Z0 to X2 Y2 Z1
   in point X2 Y2 Z1
            transmission of force in Y direction released
            transmission of moment around Z axis released

RELEASES IN: column 4

   released in segment from point X0 Y3 Z0 to X0 Y3 Z1
   in point X0 Y3 Z1
            transmission of force in X direction released
            transmission of force in Y direction released
            transmission of moment around Z axis released

RELEASES IN: column 5

   released in segment from point X2 Y3 Z0 to X2 Y3 Z1
   in point X2 Y3 Z1
            transmission of force in Y direction released
            transmission of moment around Z axis released
```

The above releases, made during the evaluation process, cause the distribution of the reactions at the supports to be as illustrated in Figure 7-7.

The above releases made by EVAL create a severe torsion problem in the first floor since the wall remains as the element responsible for the torsional stability of the first floor. The results of the evaluation of torsion in horizontal subsystem 1 are summarized below:

```
Center of gravity 1 -> Xm =       5.00 [M]    Ym =       2.50 [M]
Center of torsion 1 -> Xt =       1.25 [M]    Yt =       5.00 [M]

Elements with a significant contribution to the torsional stability of hori-
zontal subsystem 1 are:

  1) wall_xx   1, contributing with -97.31% of the torsional resistance.
```

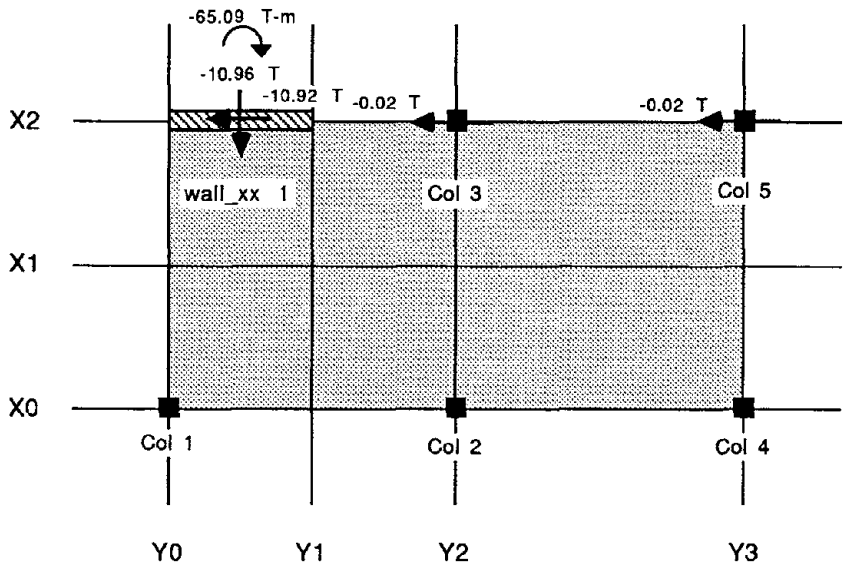


Figure 7-7: Reactions at the supports in the third evaluation example

SECTION 8
CONCLUSIONS

This section presents a summary and an evaluation of the work presented. The section also describes the contributions made by this work in the areas of computer-aided engineering and seismic evaluation of existing structures and suggests further work in these two areas.

8.1 Summary

In summary, this study addressed the following issues.

- The need of customizing engineering processes. The case was made for the personalization of engineering tasks in which there is no consensus about the *correct* way of solving these tasks. It was also argued that in engineering, the processes that require customization also include common knowledge of the discipline of interest that is not subject to customization. A framework for expert systems for domains in which there exists a combination of common and custom knowledge was proposed.

- The need for a *user friendly* representation of custom knowledge. A representation for custom knowledge in the form of decision tables was presented and detailed. A facility for creating and maintaining these decision tables was developed.

- An extensive application of the proposed framework. The proposed framework for expert systems and the facility for acquiring custom knowledge were used to implement EVAL, an expert system for evaluating the seismic resistance of existing buildings. The implementation details of EVAL and the description of the core knowledge base, an initial default custom knowledge base, as well as the interaction of these knowledge bases, was presented.

8.2 Evaluation

The need for customization of some engineering KBES applications, presented and illustrated in Section 3, is beyond any doubt. This need is present in any application that involves subjective processes and intended to be used simultaneously by multiple users or groups of users that may have different opinions, needs, or project conditions and/or goals. The customization of applications is likely to affect only some aspects of the application of interest due to the existence of common knowledge in the domain of the application that is not subject to individual interpretation. The taxonomy of knowledge in an application is not a trivial task and it, in turn, may be a highly subjective procedure.

The proposed framework for the development of KBES relies on the classification of the knowledge contained in the KBES into: commonly agreed and custom or individual knowledge. The fundamental idea of the proposed approach is to identify the subjective knowledge in a domain in order to allow the different user organizations to define and modify in the KBES the knowledge previously classified as subjective. The success of the proposed approach relies on:

- a proper classification of the knowledge to be represented in the KBES (common vs custom knowledge);
- the selection of an adequate representation for custom knowledge; and
- a proper knowledge acquisition facility that allows individual users to review and modify their custom knowledge; a key component of the knowledge acquisition facility is the user interface, which must be as *user friendly* as possible to facilitate the access to the custom knowledge of the KBES by non-computer specialists.

The modification of the custom knowledge base of the KBES should be restricted to the relevant individuals within an organization. Failure to do so could lead to improper customization and eventually to incorrect results.

As it was mentioned above, one of the principal issues in the design and implementation of KBES with the customization characteristics described in this work is the selection of the representation scheme for custom knowledge, as well as the design of the interaction between custom and core knowledge bases. These observations are made from the experience of developing and implementing EVAL. Decision tables proved to be a valuable representation scheme for custom knowledge by their ability of grouping related rules and incorporating *ELSE* rules that complete the logic. The capability of having different rule-firing strategies also proved to be very useful, since it increases the range of possibilities of expressing logic in the decision tables. Depending on the nature of the application, additional rule firing strategies could be developed for selecting the rule(s) to be executed in a decision table.

The seismic assessment of existing buildings is a domain that is clearly subject to customization. The coexistence of commonly agreed and custom knowledge in this domain is clear, since the domain includes knowledge from mature disciplines, such as structural analysis and mechanics, as well as heuristics, such as the interpretation of results from analytical procedures or the derivation of missing or incomplete information about a building. Furthermore, the heuristic knowledge in the domain of seismic assessment of existing buildings is regional and highly subjective, as structural engineering practices as well as the seismic characteristics vary from region to region. The characteristics of custom knowledge in the domain of seismic assessment of existing buildings support the fact that a KBES in this domain with a single knowledge base will not be consistent with the experience and practice of different domain experts. Thus, an application in this domain must be subject to customization.

The current heuristics in EVAL are an initial approximation of a more elaborate system that could include knowledge about several types of typical structural systems. The current version of EVAL is a computer tool aimed at a narrow group of buildings: reinforced concrete structures, using crude approximations for inferring the missing parameters relevant to the problem and for computing the capacities of individual structural members. EVAL in its current state does not incorporate heuristic knowledge from a practitioner, since this knowledge was not available. The key ingredient of any KBES is the knowledge it embodies, and EVAL could be enriched considerably with opinions of domain experts.

The information produced by the evaluation reports of EVAL provide useful hints to determine remedial methods for a potential problem diagnosed in EVAL. Although EVAL does not directly presents remedial methods, it produces information that can direct the user to select upgrading methods. In the case of the evaluation of torsion, the causes of a significant torsion problem can include either a concentration of mass or stiffness in a particular location of the building. There are several options for reducing torsion in a building, including the placement of new members or the relocation of concentrated masses. The information presented in the evaluation report for torsion gives the user an idea of the torsional characteristics of the layout of the building being evaluated. This information can be used as a basis for selecting a strategy for reducing torsion.

8.3 Contributions

This study has attempted to provide a contribution to an aspect of computer-aided engineering (CAE) that has not been formally addressed previously: the provision of a CAE framework that accommodates the coexistence of common and custom knowledge.

The conceptual framework for customizable expert systems presented responds to the dual need of providing simultaneously for specialization by individual users and for potential generalization by the discipline at large. Every domain contains several core knowledge base components that can serve as basic tools for expert system development in that domain. The development of computer-aided tools in a particular domain using the approach described in this work can lead to a better understanding of that domain by defining the roles of the different processes involved in the overall solution process. By understanding the roles of the different processes in one domain, consensus may be achieved in defining what is commonly agreed and what is custom knowledge in that particular domain. The approach of identifying core knowledge bases may also result in the rapid prototyping of not just one application in the domain but an entire set of related applications. This set of applications could have overlapping but not identical information-processing needs.

EVAL is a response to a major need. The seismic evaluation of existing buildings is a major issue in several regions of the United States and other countries. The large stock of buildings that need to be evaluated for seismic safety makes it necessary to continue the development of tools such as EVAL.

EVAL differs from previous KBES in the domain of assessing the seismic safety of existing buildings in that is implemented with the framework for KBES described in this work. No other KBES in the domain of earthquake engineering has attempted to permit individual users the possibility of expressing and modifying their subjective knowledge.

8.4 Suggestions for Further Work

This section presents suggestions for further work on the current implementation of the proposed framework.

8.4.1 Framework

A useful enhancement to the current knowledge acquisition facility of EVAL, Eval-KAF, would be to allow linking parameters of knowledge sources to a database to derive their values using, say, SQL commands. This capability can be introduced by using one of the available commercial database packages that offer a C interface, or by developing a relational database with a query language having the characteristics of a production system.

Currently, Eval-KAF, *compiles* the decision tables in the custom knowledge base into C functions, which need to be compiled and linked to the rest of the application. Eval-KAF uses the *rule mask* method for evaluating the decision tables. This method requires the use of several mask matrices for evaluating a decision table, which Eval-KAF loads from a disk file. If speed of execution would be an issue, it would be better not to use the rule mask method and use instead a method that generates a decision tree from a decision table. The use of such methods will avoid the use of mask matrices and will also reduce the number of condition tests since a rule may be executed without having to evaluate all the conditions.

The use of compiled C functions that need to be linked with the rest of the application eliminates the possibility of selecting the knowledge sources to be used at run-time. In some applications, it may be desirable to load and/or select the knowledge sources at run-time. In this case, it would be more desirable to use an interpreter of decision tables, rather than a compiler. This interpreter would have to use the rule masking method or any other method that allows the interpretation of decision tables.

Another useful enhancement to Eval-KAF would be to allow more than one choice of custom knowledge representation. This enhancement would enrich the possibilities to represent individual preferences or expertise. Currently, only decision tables are being used, but the list could be expanded to incorporate other representations such as personal constructs or rating grids. Similarly, the capability of declaring imperatively a series of actions could be added to Eval-KAF; this capability would permit the user to define the execution of a sequence of actions without imposing any conditions to determine their execution.

With respect to the interaction of core and custom knowledge bases, in EVAL, the processes in the core knowledge base are written in OPS83 and operate on a context represented in O-A-V (object-attribute-value) scheme. The custom knowledge base is implemented as a collection of C functions that operate with a set of parameters that must be passed to the function. There is no direct mechanism for mapping the attributes of an object to the parameters of a knowledge source. The method described in Section 6 for interfacing the core and custom knowledge bases seemed to be the only feasible solution. The inconveniences of this method is that any alteration in the interaction between core and custom knowledge must be incorporated by hand: modifying the OPS83 rule that gathers the necessary attributes to be passed to the knowledge source, and/or the call made to the C function that implements the knowledge source. Mapping between attributes of working memory elements (OPS83 objects) and decision table parameters has also to be taken care by hand. The use of another rule-based environment, such as a custom-made rule interpreter, could improve the issue of mapping attributes from the context of

the expert system to the knowledge sources. However, OPS83 offers the advantage of having the pattern-matching capabilities built in and allowing the user to define custom rule-matching routines.

8.4.2 Domain

The domain knowledge contained in the current version of EVAL is only a small subset of the domain of seismic assessment of existing buildings. A great deal of effort was spent on the implementation of low-level processes, such as structural analysis and geometrical reasoning. On top of these lower level routines, more elaborate and specific knowledge of seismic evaluation can be added.

The assumption made in the current version of EVAL that the resistance of connections is associated with the strength of individual elements meeting at that connection is valid for cast-in-place concrete structures; however, this hypothesis is not true in other types of structures, such as steel, masonry, or wood structures, where the connection between members is a separate entity with its own specific resistance.

Although the domain knowledge in the current version of EVAL is limited, EVAL in its current state can point out relevant potential problems in a building, such as excessive torsion, lack of sufficient redundancy, and excessive internal forces in structural members. The ability to include member releases is a useful *what-if* capability that gives an illustrative idea of the effect that the failure of a particular member would have in the seismic behavior of a building. A useful enhancement to the current version of EVAL would be to include releases of force components in the local coordinate system. This will eliminate the current approximation used in EVAL to determine releases in the global coordinate system.

SECTION 9
REFERENCES

[1] ABK.
Methodology for Mitigation of Seismic Hazards in Existing Unreinforced Masonry Buildings: The Methodology.
Technical Report 08, A Joint Venture of Agbabian Associates, S. B. Barnes and Associates, and Kariotis and Associates (ABK), Los Angeles, California, 1984.

[2] Applied Technology Council.
ATC-14: Evaluating the Seismic Resistance of Existing Buildings.
National Science Foundation, 1987.

[3] Applied Technology Council.
ATC-3-6: Tentative Provisions for the Development of Seismic Regulations for Buildings.
National Science Foundation, 1978.

[4] Bazan Enrique, Meli Roberto.
Manual de Diseno Sismico de Estructuras de acuerdo al Reglamento de Construcciones para el Distrito Federal.
Instituto de Ingenieria, UNAM, Ciudad Universitaria, Mexico, 1983.

[5] Becker, Sue and Selman Bart.
An Overview of Knowledge Acquisition Methods for Expert Systems.
Technical Report CSRI-184, Computer Systems Research Institute, University of Toronto, June 1986.

[6] Benett, J. S.
Roget: Acquiring the Conceptual Structure of a Diagnostic Expert System.
In *Fourth National Conference on Entity-Relationship Approach.* IEEE, October 1985.

[7] Boose, John H.
Expertise Transfer for Expert System Design.
elsevier, 1986.

[8] Brekke, R. L.
Benchmarking Expert System Tool Performance.
Ford Aerospace and Communications Corp., Space Information Systems Operation , September 1982.

[9] Davis, R., and Lenat D.
Knowledge-Based Systems in Artificial Intelligence.
McGraw-Hill, 1982.

[10] de Greef, P. and Lenat, D.
A Case Study in Structured Knowedge Acquisition.
In *Seventh International Joint Conference on Artificial Intelligence.* Los Angeles, CA, August 1985.

[11] Dym, Clive L., and Levitt, Raymond E.
Knowledge-Based Systems in Engineering.
McGraw-Hill Book Company, New York, NY, 1990.

[12] 12-743 Expert Systems in Civil Engineering Course Notes.
Department of Civil Engineering, Carnegie Mellon University.
Spring 1990.

[13] Fenves, S.J., Wright, R., Stahl, F. and Reed K.
Introduction to SASE: Standards Analysis, Synthesis and Expression.
U.S. Department of Commerce, Gaithersburgh, MD, 1987.

[14] Fenves, Steven J.
Tabular Decision Logic for Structural Design.
Journal of the Structural Division , 1966.
ASCE, Vol. 92.

[15] Forgy, C. L.
OPS5 User's manual.
Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie Mellon University, 1981.

[16] Forgy, C. L.
Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem.
Artificial Intelligence , pp. 17-37, September 1982.

[17] Fu, K.S., and Yao, J. T. P.
Speril: An Expert System for Safety Evaluation of Structures.
In *Proceedings of the IEEE conference on systems, man and cybernetics.* Institute of Electrical and Electronic Engineers, 1984.

[18] Genesereth, Michael R., and Ginsberg, Matthew L.
Logic programming.
Communications of the ACM , September 1985.

[19] Glasgow, B., and Graham, E.
Rapid Prototyping Using Core Knowledge Bases.
AI Expert , May 1988.

[20] Gonzalez-Cuevas, O, Robles-F.V., F, Casillas, J, Diaz-de-Cossio, R.
Aspectos Fundamentales del Concreto Reforzado.
Limusa S.A., Mexico D.F., 1982.

[21] Grover, M.D.
A Pragmatic Knowledge Acquisition Methodology.
In *Eight International Joint Conference on Artificial Intelligence.* Karlsruhe, West Germany, August 1983.

[22] Harmon, Paul, and King, David.
Artificial Intelligence in Business. Expert Systems.
Wiley Press, 1985.

[23] Hayes-Roth, F., Waterman, D., and Lenat,D.
Building Expert Systems.
Addison-Wesley, 1983.

[24] Hearn, D., and Baker, M. P.
Computer Graphics.
Prentice Hall Inc., 1986.

[25] Hsu, T.C.C.
Torsion of structural concrete sections.
Technical Report ACI-SP-18, American Concrete Institute, 1968.

[26] Ibarra-Anaya, Enrique.
Analisis Sismico Dinamico de Marcos Planos Utilizando Microcomputadora.
Facultad de Ingenieria, UNAM, Ciudad Universitaria, Mexico D.F., 1986.

[27] Ibarra-Anaya, E., and Fenves, S. J.
Technical Description of the Framework for Customizable Knowledge-Based Expert Systems.
Technical Report R-90-197, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, October 1990.

[28] Ishizuka, M., Eu, K. S., and Yao, J. T. P.
Speril: An Expert System for Damage Assessment of Existing Structures.
In *Proceedings of the 6th international conference on pattern recognition.* Institute of Electrical and Electronic Engineers, 1982.

[29] James, Martin, and Oxman, Steven.
Building Expert Systems. A Tutorial.
Prentice-Hall, 1988.

[30] Kelly, George Alexander.
The Psychology of Personal Constructs.
W. W. Norton, 1955.

[31] Kelly, George Alexander.
A Theory of Personality; the psychology of personal constructs.
W. W. Norton, 1963.

[32] Kelly, George Alexander.
Clinical Psychology and Personality; the selected papers of George Kelly. Edited by Brendan Maher.
Wiley, 1969.

[33] Kirk, H.W.
Use of Decision Tables in Computer Programming.
Communications ACM, Vol. 8 , 1965.

[34] Leaman, Claire M.
Rule-Based Structural Design in C.
AI Expert , May 1989.

[35] Marcus, Sandra.
Automating Knowledge Acquisition Facility for Expert Systems.
Kluwer Academic Publisher, 101 Philip Drive, Assinippi Park, Norwell, Massachussets 02061, 1988.

[36] Marcus S., McDermott, J., and Wang, T.
Knowledge Acquisition for Constructive Systems.
In *IJCAI-85.* IJCAI, Los Angeles, CA, October 1985.

[37] Mauch, S. P., and Fenves, S.J.
Releases and Constraints in Structural Networks.
In *Journal of the Structural Division.* ASCE, October 1967.

[38] Mijango-Penate, Carolina de Lourdes.
ASALVO: A Knowledge-Based Expert System for the Seismic Safety Assesment of Existing Structures.
Master's thesis, Department of Civil Engineering, Massachusets Institute of Technology, 1988.

[39] Peters, Joseph Francis.
SEES: An Expert System for the Strength Evaluation of Existing Structural Members.
Master's thesis, Department of Civil Engineering, Carnegie Mellon University, 1987.

[40] Reboh, R.
The Knowledge Acquisition System (PROSPECTOR: A Computer-Based Consultant for Mineral Exploration).
Artificial Intelligence Center, SRI International, Menlo Park, CA, Sept 1979.

[41] Sedgewick, Robert.
Algorithms in C.
Addison-Wesley, 1990.

[42] Shaw, M. L. G.
On Becoming a Personal Scientist.
Academic Press, 1980.

[43] Siller, T. J., Fenves, S. J., Bielak, J., and Reed, D.
A Prototype Knowledge-Based System for Seismic Damage Assessment.
Technical Report R-89-177, Department of Civil Engineering, Carnegie Mellon University, March, 1989.
NCEER project number 86-4022.

[44] Tazir, Zabra el Hayat.
Amadeus: Advisory Methodology for the Assessment of Damage after Earthquake and Usability Structures.
Unpublished paper, Department of Civil Engineering, Massachusets Institute of Technology, 1987.

[45] Welland R.
Decision Tables and Computer Programming.
Heyden & Son Ltd, London, UK, 1981.

[46] Gergeley, P., Abel, J. F., Conley, C. H., Wilson, J. L., Mueller, P., Fenves, S. J., Bielak, J.
Expert Systems in Earthquake-Resistant Design of Buildings.
In *Workshop on expert systems in earthquake resistant design of buildings.* Cornell University, 1989.
NCEER projects 871009 and 876001.

[47] Zozaya-Gorostiza, C., Hendrickson, C., and Rehak D. R.
Knowledge-Based Process Planning for Construction and Manufacturing.
Academic Press, Inc., 1989.

NATIONAL CENTER FOR EARTHQUAKE ENGINEERING RESEARCH
LIST OF TECHNICAL REPORTS

The National Center for Earthquake Engineering Research (NCEER) publishes technical reports on a variety of subjects related to earthquake engineering written by authors funded through NCEER. These reports are available from both NCEER's Publications Department and the National Technical Information Service (NTIS). Requests for reports should be directed to the Publications Department, National Center for Earthquake Engineering Research, State University of New York at Buffalo, Red Jacket Quadrangle, Buffalo, New York 14261. Reports can also be requested through NTIS, 5285 Port Royal Road, Springfield, Virginia 22161. NTIS accession numbers are shown in parenthesis, if available.

NCEER-87-0001 "First-Year Program in Research, Education and Technology Transfer," 3/5/87, (PB88-134275/AS).

NCEER-87-0002 "Experimental Evaluation of Instantaneous Optimal Algorithms for Structural Control," by R.C. Lin, T.T. Soong and A.M. Reinhorn, 4/20/87, (PB88-134341/AS).

NCEER-87-0003 "Experimentation Using the Earthquake Simulation Facilities at University at Buffalo," by A.M. Reinhorn and R.L. Ketter, to be published.

NCEER-87-0004 "The System Characteristics and Performance of a Shaking Table," by J.S. Hwang, K.C. Chang and G.C. Lee, 6/1/87, (PB88-134259/AS). This report is available only through NTIS (see address given above).

NCEER-87-0005 "A Finite Element Formulation for Nonlinear Viscoplastic Material Using a Q Model," by O. Gyebi and G. Dasgupta, 11/2/87, (PB88-213764/AS).

NCEER-87-0006 "Symbolic Manipulation Program (SMP) - Algebraic Codes for Two and Three Dimensional Finite Element Formulations," by X. Lee and G. Dasgupta, 11/9/87, (PB88-219522/AS).

NCEER-87-0007 "Instantaneous Optimal Control Laws for Tall Buildings Under Seismic Excitations," by J.N. Yang, A. Akbarpour and P. Ghaemmaghami, 6/10/87, (PB88-134333/AS).

NCEER-87-0008 "IDARC: Inelastic Damage Analysis of Reinforced Concrete Frame - Shear-Wall Structures," by Y.J. Park, A.M. Reinhorn and S.K. Kunnath, 7/20/87, (PB88-134325/AS).

NCEER-87-0009 "Liquefaction Potential for New York State: A Preliminary Report on Sites in Manhattan and Buffalo," by M. Budhu, V. Vijayakumar, R.F. Giese and L. Baumgras, 8/31/87, (PB88-163704/AS). This report is available only through NTIS (see address given above).

NCEER-87-0010 "Vertical and Torsional Vibration of Foundations in Inhomogeneous Media," by A.S. Veletsos and K.W. Dotson, 6/1/87, (PB88-134291/AS).

NCEER-87-0011 "Seismic Probabilistic Risk Assessment and Seismic Margins Studies for Nuclear Power Plants," by Howard H.M. Hwang, 6/15/87, (PB88-134267/AS).

NCEER-87-0012 "Parametric Studies of Frequency Response of Secondary Systems Under Ground-Acceleration Excitations," by Y. Yong and Y.K. Lin, 6/10/87, (PB88-134309/AS).

NCEER-87-0013 "Frequency Response of Secondary Systems Under Seismic Excitation," by J.A. HoLung, J. Cai and Y.K. Lin, 7/31/87, (PB88-134317/AS).

NCEER-87-0014 "Modelling Earthquake Ground Motions in Seismically Active Regions Using Parametric Time Series Methods," by G.W. Ellis and A.S. Cakmak, 8/25/87, (PB88-134283/AS).

NCEER-87-0015 "Detection and Assessment of Seismic Structural Damage," by E. DiPasquale and A.S. Cakmak, 8/25/87, (PB88-163712/AS).

NCEER-87-0016 "Pipeline Experiment at Parkfield, California," by J. Isenberg and E. Richardson, 9/15/87, (PB88-163720/AS). This report is available only through NTIS (see address given above).

NCEER-87-0017 "Digital Simulation of Seismic Ground Motion," by M. Shinozuka, G. Deodatis and T. Harada, 8/31/87, (PB88-155197/AS). This report is available only through NTIS (see address given above).

NCEER-87-0018 "Practical Considerations for Structural Control: System Uncertainty, System Time Delay and Truncation of Small Control Forces," J.N. Yang and A. Akbarpour, 8/10/87, (PB88-163738/AS).

NCEER-87-0019 "Modal Analysis of Nonclassically Damped Structural Systems Using Canonical Transformation," by J.N. Yang, S. Sarkani and F.X. Long, 9/27/87, (PB88-187851/AS).

NCEER-87-0020 "A Nonstationary Solution in Random Vibration Theory," by J.R. Red-Horse and P.D. Spanos, 11/3/87, (PB88-163746/AS).

NCEER-87-0021 "Horizontal Impedances for Radially Inhomogeneous Viscoelastic Soil Layers," by A.S. Veletsos and K.W. Dotson, 10/15/87, (PB88-150859/AS).

NCEER-87-0022 "Seismic Damage Assessment of Reinforced Concrete Members," by Y.S. Chung, C. Meyer and M. Shinozuka, 10/9/87, (PB88-150867/AS). This report is available only through NTIS (see address given above).

NCEER-87-0023 "Active Structural Control in Civil Engineering," by T.T. Soong, 11/11/87, (PB88-187778/AS).

NCEER-87-0024 Vertical and Torsional Impedances for Radially Inhomogeneous Viscoelastic Soil Layers," by K.W. Dotson and A.S. Veletsos, 12/87, (PB88-187786/AS).

NCEER-87-0025 "Proceedings from the Symposium on Seismic Hazards, Ground Motions, Soil-Liquefaction and Engineering Practice in Eastern North America," October 20-22, 1987, edited by K.H. Jacob, 12/87, (PB88-188115/AS).

NCEER-87-0026 "Report on the Whittier-Narrows, California, Earthquake of October 1, 1987," by J. Pantelic and A. Reinhorn, 11/87, (PB88-187752/AS). This report is available only through NTIS (see address given above).

NCEER-87-0027 "Design of a Modular Program for Transient Nonlinear Analysis of Large 3-D Building Structures," by S. Srivastav and J.F. Abel, 12/30/87, (PB88-187950/AS).

NCEER-87-0028 "Second-Year Program in Research, Education and Technology Transfer," 3/8/88, (PB88-219480/AS).

NCEER-88-0001 "Workshop on Seismic Computer Analysis and Design of Buildings With Interactive Graphics," by W. McGuire, J.F. Abel and C.H. Conley, 1/18/88, (PB88-187760/AS).

NCEER-88-0002 "Optimal Control of Nonlinear Flexible Structures," by J.N. Yang, F.X. Long and D. Wong, 1/22/88, (PB88-213772/AS).

NCEER-88-0003 "Substructuring Techniques in the Time Domain for Primary-Secondary Structural Systems," by G.D. Manolis and G. Juhn, 2/10/88, (PB88-213780/AS).

NCEER-88-0004 "Iterative Seismic Analysis of Primary-Secondary Systems," by A. Singhal, L.D. Lutes and P.D. Spanos, 2/23/88, (PB88-213798/AS).

NCEER-88-0005 "Stochastic Finite Element Expansion for Random Media," by P.D. Spanos and R. Ghanem, 3/14/88, (PB88-213806/AS).

NCEER-88-0006 "Combining Structural Optimization and Structural Control," by F.Y. Cheng and C.P. Pantelides, 1/10/88, (PB88-213814/AS).

NCEER-88-0007 "Seismic Performance Assessment of Code-Designed Structures," by H.H-M. Hwang, J-W. Jaw and H-J. Shau, 3/20/88, (PB88-219423/AS).

NCEER-88-0008 "Reliability Analysis of Code-Designed Structures Under Natural Hazards," by H.H-M. Hwang, H. Ushiba and M. Shinozuka, 2/29/88, (PB88-229471/AS).

NCEER-88-0009 "Seismic Fragility Analysis of Shear Wall Structures," by J-W Jaw and H.H-M. Hwang, 4/30/88, (PB89-102867/AS).

NCEER-88-0010 "Base Isolation of a Multi-Story Building Under a Harmonic Ground Motion - A Comparison of Performances of Various Systems," by F-G Fan, G. Ahmadi and I.G. Tadjbakhsh, 5/18/88, (PB89-122238/AS).

NCEER-88-0011 "Seismic Floor Response Spectra for a Combined System by Green's Functions," by F.M. Lavelle, L.A. Bergman and P.D. Spanos, 5/1/88, (PB89-102875/AS).

NCEER-88-0012 "A New Solution Technique for Randomly Excited Hysteretic Structures," by G.Q. Cai and Y.K. Lin, 5/16/88, (PB89-102883/AS).

NCEER-88-0013 "A Study of Radiation Damping and Soil-Structure Interaction Effects in the Centrifuge," by K. Weissman, supervised by J.H. Prevost, 5/24/88, (PB89-144703/AS).

NCEER-88-0014 "Parameter Identification and Implementation of a Kinematic Plasticity Model for Frictional Soils," by J.H. Prevost and D.V. Griffiths, to be published.

NCEER-88-0015 "Two- and Three- Dimensional Dynamic Finite Element Analyses of the Long Valley Dam," by D.V. Griffiths and J.H. Prevost, 6/17/88, (PB89-144711/AS).

NCEER-88-0016 "Damage Assessment of Reinforced Concrete Structures in Eastern United States," by A.M. Reinhorn, M.J. Seidel, S.K. Kunnath and Y.J. Park, 6/15/88, (PB89-122220/AS).

NCEER-88-0017 "Dynamic Compliance of Vertically Loaded Strip Foundations in Multilayered Viscoelastic Soils," by S. Ahmad and A.S.M. Israil, 6/17/88, (PB89-102891/AS).

NCEER-88-0018 "An Experimental Study of Seismic Structural Response With Added Viscoelastic Dampers," by R.C. Lin, Z. Liang, T.T. Soong and R.H. Zhang, 6/30/88, (PB89-122212/AS).

NCEER-88-0019 "Experimental Investigation of Primary - Secondary System Interaction," by G.D. Manolis, G. Juhn and A.M. Reinhorn, 5/27/88, (PB89-122204/AS).

NCEER-88-0020 "A Response Spectrum Approach For Analysis of Nonclassically Damped Structures," by J.N. Yang, S. Sarkani and F.X. Long, 4/22/88, (PB89-102909/AS).

NCEER-88-0021 "Seismic Interaction of Structures and Soils: Stochastic Approach," by A.S. Veletsos and A.M. Prasad, 7/21/88, (PB89-122196/AS).

NCEER-88-0022 "Identification of the Serviceability Limit State and Detection of Seismic Structural Damage," by E. DiPasquale and A.S. Cakmak, 6/15/88, (PB89-122188/AS).

NCEER-88-0023 "Multi-Hazard Risk Analysis: Case of a Simple Offshore Structure," by B.K. Bhartia and E.H. Vanmarcke, 7/21/88, (PB89-145213/AS).

NCEER-88-0024 "Automated Seismic Design of Reinforced Concrete Buildings," by Y.S. Chung, C. Meyer and M. Shinozuka, 7/5/88, (PB89-122170/AS).

NCEER-88-0025 "Experimental Study of Active Control of MDOF Structures Under Seismic Excitations," by L.L. Chung, R.C. Lin, T.T. Soong and A.M. Reinhorn, 7/10/88, (PB89-122600/AS).

NCEER-88-0026 "Earthquake Simulation Tests of a Low-Rise Metal Structure," by J.S. Hwang, K.C. Chang, G.C. Lee and R.L. Ketter, 8/1/88, (PB89-102917/AS).

NCEER-88-0027 "Systems Study of Urban Response and Reconstruction Due to Catastrophic Earthquakes," by F. Kozin and H.K. Zhou, 9/22/88, (PB90-162348/AS).

NCEER-88-0028 "Seismic Fragility Analysis of Plane Frame Structures," by H.H-M. Hwang and Y.K. Low, 7/31/88, (PB89-131445/AS).

NCEER-88-0029 "Response Analysis of Stochastic Structures," by A. Kardara, C. Bucher and M. Shinozuka, 9/22/88, (PB89-174429/AS).

NCEER-88-0030 "Nonnormal Accelerations Due to Yielding in a Primary Structure," by D.C.K. Chen and L.D. Lutes, 9/19/88, (PB89-131437/AS).

NCEER-88-0031 "Design Approaches for Soil-Structure Interaction," by A.S. Veletsos, A.M. Prasad and Y. Tang, 12/30/88, (PB89-174437/AS).

NCEER-88-0032 "A Re-evaluation of Design Spectra for Seismic Damage Control," by C.J. Turkstra and A.G. Tallin, 11/7/88, (PB89-145221/AS).

NCEER-88-0033 "The Behavior and Design of Noncontact Lap Splices Subjected to Repeated Inelastic Tensile Loading," by V.E. Sagan, P. Gergely and R.N. White, 12/8/88, (PB89-163737/AS).

NCEER-88-0034 "Seismic Response of Pile Foundations," by S.M. Mamoon, P.K. Banerjee and S. Ahmad, 11/1/88, (PB89-145239/AS).

NCEER-88-0035 "Modeling of R/C Building Structures With Flexible Floor Diaphragms (IDARC2)," by A.M. Reinhorn, S.K. Kunnath and N. Panahshahi, 9/7/88, (PB89-207153/AS).

NCEER-88-0036 "Solution of the Dam-Reservoir Interaction Problem Using a Combination of FEM, BEM with Particular Integrals, Modal Analysis, and Substructuring," by C-S. Tsai, G.C. Lee and R.L. Ketter, 12/31/88, (PB89-207146/AS).

NCEER-88-0037 "Optimal Placement of Actuators for Structural Control," by F.Y. Cheng and C.P. Pantelides, 8/15/88, (PB89-162846/AS).

NCEER-88-0038 "Teflon Bearings in Aseismic Base Isolation: Experimental Studies and Mathematical Modeling," by A. Mokha, M.C. Constantinou and A.M. Reinhorn, 12/5/88, (PB89-218457/AS).

NCEER-88-0039 "Seismic Behavior of Flat Slab High-Rise Buildings in the New York City Area," by P. Weidlinger and M. Ettouney, 10/15/88, (PB90-145681/AS).

NCEER-88-0040 "Evaluation of the Earthquake Resistance of Existing Buildings in New York City," by P. Weidlinger and M. Ettouney, 10/15/88, to be published.

NCEER-88-0041 "Small-Scale Modeling Techniques for Reinforced Concrete Structures Subjected to Seismic Loads," by W. Kim, A. El-Attar and R.N. White, 11/22/88, (PB89-189625/AS).

NCEER-88-0042 "Modeling Strong Ground Motion from Multiple Event Earthquakes," by G.W. Ellis and A.S. Cakmak, 10/15/88, (PB89-174445/AS).

NCEER-88-0043 "Nonstationary Models of Seismic Ground Acceleration," by M. Grigoriu, S.E. Ruiz and E. Rosenblueth, 7/15/88, (PB89-189617/AS).

NCEER-88-0044 "SARCF User's Guide: Seismic Analysis of Reinforced Concrete Frames," by Y.S. Chung, C. Meyer and M. Shinozuka, 11/9/88, (PB89-174452/AS).

NCEER-88-0045 "First Expert Panel Meeting on Disaster Research and Planning," edited by J. Pantelic and J. Stoyle, 9/15/88, (PB89-174460/AS).

NCEER-88-0046 "Preliminary Studies of the Effect of Degrading Infill Walls on the Nonlinear Seismic Response of Steel Frames," by C.Z. Chrysostomou, P. Gergely and J.F. Abel, 12/19/88, (PB89-208383/AS).

NCEER-88-0047 "Reinforced Concrete Frame Component Testing Facility - Design, Construction, Instrumentation and Operation," by S.P. Pessiki, C. Conley, T. Bond, P. Gergely and R.N. White, 12/16/88, (PB89-174478/AS).

NCEER-89-0001 "Effects of Protective Cushion and Soil Compliancy on the Response of Equipment Within a Seismically Excited Building," by J.A. HoLung, 2/16/89, (PB89-207179/AS).

NCEER-89-0002 "Statistical Evaluation of Response Modification Factors for Reinforced Concrete Structures," by H.H-M. Hwang and J-W. Jaw, 2/17/89, (PB89-207187/AS).

NCEER-89-0003 "Hysteretic Columns Under Random Excitation," by G-Q. Cai and Y.K. Lin, 1/9/89, (PB89-196513/AS).

NCEER-89-0004 "Experimental Study of 'Elephant Foot Bulge' Instability of Thin-Walled Metal Tanks," by Z-H. Jia and R.L. Ketter, 2/22/89, (PB89-207195/AS).

NCEER-89-0005 "Experiment on Performance of Buried Pipelines Across San Andreas Fault," by J. Isenberg, E. Richardson and T.D. O'Rourke, 3/10/89, (PB89-218440/AS).

NCEER-89-0006 "A Knowledge-Based Approach to Structural Design of Earthquake-Resistant Buildings," by M. Subramani, P. Gergely, C.H. Conley, J.F. Abel and A.H. Zaghw, 1/15/89, (PB89-218465/AS).

NCEER-89-0007 "Liquefaction Hazards and Their Effects on Buried Pipelines," by T.D. O'Rourke and P.A. Lane, 2/1/89, (PB89-218481).

NCEER-89-0008 "Fundamentals of System Identification in Structural Dynamics," by H. Imai, C-B. Yun, O. Maruyama and M. Shinozuka, 1/26/89, (PB89-207211/AS).

NCEER-89-0009 "Effects of the 1985 Michoacan Earthquake on Water Systems and Other Buried Lifelines in Mexico," by A.G. Ayala and M.J. O'Rourke, 3/8/89, (PB89-207229/AS).

NCEER-89-R010 "NCEER Bibliography of Earthquake Education Materials," by K.E.K. Ross, Second Revision, 9/1/89, (PB90-125352/AS).

NCEER-89-0011 "Inelastic Three-Dimensional Response Analysis of Reinforced Concrete Building Structures (IDARC-3D), Part I - Modeling," by S.K. Kunnath and A.M. Reinhorn, 4/17/89, (PB90-114612/AS).

NCEER-89-0012 "Recommended Modifications to ATC-14," by C.D. Poland and J.O. Malley, 4/12/89, (PB90-108648/AS).

NCEER-89-0013 "Repair and Strengthening of Beam-to-Column Connections Subjected to Earthquake Loading," by M. Corazao and A.J. Durrani, 2/28/89, (PB90-109885/AS).

NCEER-89-0014 "Program EXKAL2 for Identification of Structural Dynamic Systems," by O. Maruyama, C-B. Yun, M. Hoshiya and M. Shinozuka, 5/19/89, (PB90-109877/AS).

NCEER-89-0015 "Response of Frames With Bolted Semi-Rigid Connections, Part I - Experimental Study and Analytical Predictions," by P.J. DiCorso, A.M. Reinhorn, J.R. Dickerson, J.B. Radziminski and W.L. Harper, 6/1/89, to be published.

NCEER-89-0016 "ARMA Monte Carlo Simulation in Probabilistic Structural Analysis," by P.D. Spanos and M.P. Mignolet, 7/10/89, (PB90-109893/AS).

NCEER-89-P017 "Preliminary Proceedings from the Conference on Disaster Preparedness - The Place of Earthquake Education in Our Schools," Edited by K.E.K. Ross, 6/23/89.

NCEER-89-0017 "Proceedings from the Conference on Disaster Preparedness - The Place of Earthquake Education in Our Schools," Edited by K.E.K. Ross, 12/31/89, (PB90-207895).

NCEER-89-0018 "Multidimensional Models of Hysteretic Material Behavior for Vibration Analysis of Shape Memory Energy Absorbing Devices, by E.J. Graesser and F.A. Cozzarelli, 6/7/89, (PB90-164146/AS).

NCEER-89-0019 "Nonlinear Dynamic Analysis of Three-Dimensional Base Isolated Structures (3D-BASIS)," by S. Nagarajaiah, A.M. Reinhorn and M.C. Constantinou, 8/3/89, (PB90-161936/AS).

NCEER-89-0020 "Structural Control Considering Time-Rate of Control Forces and Control Rate Constraints," by F.Y. Cheng and C.P. Pantelides, 8/3/89, (PB90-120445/AS).

NCEER-89-0021 "Subsurface Conditions of Memphis and Shelby County," by K.W. Ng, T-S. Chang and H-H.M. Hwang, 7/26/89, (PB90-120437/AS).

NCEER-89-0022 "Seismic Wave Propagation Effects on Straight Jointed Buried Pipelines," by K. Elhmadi and M.J. O'Rourke, 8/24/89, (PB90-162322/AS).

NCEER-89-0023 "Workshop on Serviceability Analysis of Water Delivery Systems," edited by M. Grigoriu, 3/6/89, (PB90-127424/AS).

NCEER-89-0024 "Shaking Table Study of a 1/5 Scale Steel Frame Composed of Tapered Members," by K.C. Chang, J.S. Hwang and G.C. Lee, 9/18/89, (PB90-160169/AS).

NCEER-89-0025 "DYNA1D: A Computer Program for Nonlinear Seismic Site Response Analysis - Technical Documentation," by Jean H. Prevost, 9/14/89, (PB90-161944/AS).

NCEER-89-0026 "1:4 Scale Model Studies of Active Tendon Systems and Active Mass Dampers for Aseismic Protection," by A.M. Reinhorn, T.T. Soong, R.C. Lin, Y.P. Yang, Y. Fukao, H. Abe and M. Nakai, 9/15/89, (PB90-173246/AS).

NCEER-89-0027 "Scattering of Waves by Inclusions in a Nonhomogeneous Elastic Half Space Solved by Boundary Element Methods," by P.K. Hadley, A. Askar and A.S. Cakmak, 6/15/89, (PB90-145699/AS).

NCEER-89-0028 "Statistical Evaluation of Deflection Amplification Factors for Reinforced Concrete Structures," by H.H.M. Hwang, J-W. Jaw and A.L. Ch'ng, 8/31/89, (PB90-164633/AS).

NCEER-89-0029 "Bedrock Accelerations in Memphis Area Due to Large New Madrid Earthquakes," by H.H.M. Hwang, C.H.S. Chen and G. Yu, 11/7/89, (PB90-162330/AS).

NCEER-89-0030 "Seismic Behavior and Response Sensitivity of Secondary Structural Systems," by Y.Q. Chen and T.T. Soong, 10/23/89, (PB90-164658/AS).

NCEER-89-0031 "Random Vibration and Reliability Analysis of Primary-Secondary Structural Systems," by Y. Ibrahim, M. Grigoriu and T.T. Soong, 11/10/89, (PB90-161951/AS).

NCEER-89-0032 "Proceedings from the Second U.S. - Japan Workshop on Liquefaction, Large Ground Deformation and Their Effects on Lifelines, September 26-29, 1989," Edited by T.D. O'Rourke and M. Hamada, 12/1/89, (PB90-209388/AS).

NCEER-89-0033 "Deterministic Model for Seismic Damage Evaluation of Reinforced Concrete Structures," by J.M. Bracci, A.M. Reinhorn, J.B. Mander and S.K. Kunnath, 9/27/89.

NCEER-89-0034 "On the Relation Between Local and Global Damage Indices," by E. DiPasquale and A.S. Cakmak, 8/15/89, (PB90-173865).

NCEER-89-0035 "Cyclic Undrained Behavior of Nonplastic and Low Plasticity Silts," by A.J. Walker and H.E. Stewart, 7/26/89, (PB90-183518/AS).

NCEER-89-0036 "Liquefaction Potential of Surficial Deposits in the City of Buffalo, New York," by M. Budhu, R. Giese and L. Baumgrass, 1/17/89, (PB90-208455/AS).

NCEER-89-0037 "A Determinstic Assessment of Effects of Ground Motion Incoherence," by A.S. Veletsos and Y. Tang, 7/15/89, (PB90-164294/AS).

NCEER-89-0038 "Workshop on Ground Motion Parameters for Seismic Hazard Mapping," July 17-18, 1989, edited by R.V. Whitman, 12/1/89, (PB90-173923/AS).

NCEER-89-0039 "Seismic Effects on Elevated Transit Lines of the New York City Transit Authority," by C.J. Costantino, C.A. Miller and E. Heymsfield, 12/26/89, (PB90-207887/AS).

NCEER-89-0040 "Centrifugal Modeling of Dynamic Soil-Structure Interaction," by K. Weissman, Supervised by J.H. Prevost, 5/10/89, (PB90-207879/AS).

NCEER-89-0041 "Linearized Identification of Buildings With Cores for Seismic Vulnerability Assessment," by I-K. Ho and A.E. Aktan, 11/1/89.

NCEER-90-0001 "Geotechnical and Lifeline Aspects of the October 17, 1989 Loma Prieta Earthquake in San Francisco," by T.D. O'Rourke, H.E. Stewart, F.T. Blackburn and T.S. Dickerman, 1/90, (PB90-208596/AS).

NCEER-90-0002 "Nonnormal Secondary Response Due to Yielding in a Primary Structure," by D.C.K. Chen and L.D. Lutes, 2/28/90.

NCEER-90-0003 "Earthquake Education Materials for Grades K-12," by K.E.K. Ross, 4/16/90.

NCEER-90-0004 "Catalog of Strong Motion Stations in Eastern North America," by R.W. Busby, 4/3/90.

NCEER-90-0005 "NCEER Strong-Motion Data Base: A User Manuel for the GeoBase Release (Version 1.0 for the Sun3)," by P. Friberg and K. Jacob, 3/31/90.

NCEER-90-0006 "Seismic Hazard Along a Crude Oil Pipeline in the Event of an 1811-1812 Type New Madrid Earthquake," by H.H.M. Hwang and C-H.S. Chen, 4/16/90.

NCEER-90-0007 "Site-Specific Response Spectra for Memphis Sheahan Pumping Station," by H.H.M. Hwang and C.S. Lee, 5/15/90.

NCEER-90-0008 "Pilot Study on Seismic Vulnerability of Crude Oil Transmission Systems," by T. Ariman, R. Dobry, M. Grigoriu, F. Kozin, M. O'Rourke, T. O'Rourke and M. Shinozuka, 5/25/90.

NCEER-90-0009 "A Program to Generate Site Dependent Time Histories: EQGEN," by G.W. Ellis, M. Srinivasan and A.S. Cakmak, 1/30/90.

NCEER-90-0010 "Active Isolation for Seismic Protection of Operating Rooms," by M.E. Talbott, Supervised by M. Shinozuka, 6/8/9.

NCEER-90-0011 "Program LINEARID for Identification of Linear Structural Dynamic Systems," by C-B. Yun and M. Shinozuka, 6/25/90.

NCEER-90-0012 "Two-Dimensional Two-Phase Elasto-Plastic Seismic Response of Earth Dams," by A.N. Yiagos, Supervised by J.H. Prevost, 6/20/90.

NCEER-90-0013 "Secondary Systems in Base-Isolated Structures: Experimental Investigation, Stochastic Response and Stochastic Sensitivity," by G.D. Manolis, G. Juhn, M.C. Constantinou and A.M. Reinhorn, 7/1/90.

NCEER-90-0014 "Seismic Behavior of Lightly-Reinforced Concrete Column and Beam-Column Joint Details," by S.P. Pessiki, C.H. Conley, P. Gergely and R.N. White, 8/22/90.

NCEER-90-0015 "Two Hybrid Control Systems for Building Structures Under Strong Earthquakes," by J.N. Yang and A. Danielians, 6/29/90.

NCEER-90-0016 "Instantaneous Optimal Control with Acceleration and Velocity Feedback," by J.N. Yang and Z. Li, 6/29/90.

NCEER-90-0017 "Reconnaissance Report on the Northern Iran Earthquake of June 21, 1990," by M. Mehrain, 10/4/90.

NCEER-90-0018 "Evaluation of Liquefaction Potential in Memphis and Shelby County," by T.S. Chang, P.S. Tang, C.S. Lee and H. Hwang, 8/10/90.

NCEER-90-0019 "Experimental and Analytical Study of a Combined Sliding Disc Bearing and Helical Steel Spring Isolation System," by M.C. Constantinou, A.S. Mokha and A.M. Reinhorn, 10/4/90.

NCEER-90-0020 "Experimental Study and Analytical Prediction of Earthquake Response of a Sliding Isolation System with a Spherical Surface," by A.S. Mokha, M.C. Constantinou and A.M. Reinhorn, 10/11/90.

NCEER-90-0021 "Dynamic Interaction Factors for Floating Pile Groups," by G. Gazetas, K. Fan, A. Kaynia and E. Kausel, 9/10/90.

NCEER-90-0022 "Evaluation of Seismic Damage Indices for Reinforced Concrete Structures," by S. Rodríguez-Gómez and A.S. Cakmak, 9/30/90.

NCEER-90-0023 "Study of Site Response at a Selected Memphis Site," by H. Desai, S. Ahmad, E.S. Gazetas and M.R. Oh, 10/11/90.

NCEER-90-0024 "A User's Guide to Strongmo: Version 1.0 of NCEER's Strong-Motion Data Access Tool for PCs and Terminals," by P.A. Friberg and C.A.T. Susch, 11/15/90.

NCEER-90-0025 "A Three-Dimensional Analytical Study of Spatial Variability of Seismic Ground Motions," by L-L. Hong and A.H.-S. Ang, 10/30/90.

NCEER-90-0026 "MUMOID User's Guide - A Program for the Identification of Modal Parameters," by S. Rodríguez-Gómez and E. DiPasquale, 9/30/90.

NCEER-90-0027 "SARCF-II User's Guide - Seismic Analysis of Reinforced Concrete Frames," by S. Rodríguez-Gómez, Y.S. Chung and C. Meyer, 9/30/90.

NCEER-90-0028 "Viscous Dampers: Testing, Modeling and Application in Vibration and Seismic Isolation," by N. Makris and M.C. Constantinou, 12/20/90.

NCEER-90-0029 "Soil Effects on Earthquake Ground Motions in the Memphis Area," by H. Hwang, C.S. Lee, K.W. Ng and T.S. Chang, 8/2/90.

NCEER-91-0001 "Proceedings from the Third Japan-U.S. Workshop on Earthquake Resistant Design of Lifeline Facilities and Countermeasures for Soil Liquefaction, December 17-19, 1990," edited by T.D. O'Rourke and M. Hamada, 2/1/91.

NCEER-91-0002 "Physical Space Solutions of Non-Proportionally Damped Systems," by M. Tong, Z. Liang and G.C. Lee, 1/15/91.

NCEER-91-0003 "Kinematic Seismic Response of Single Piles and Pile Groups," by K. Fan, G. Gazetas, A. Kaynia, E. Kausel and S. Ahmad, 1/10/91.

NCEER-91-0004 "Theory of Complex Damping," by Z. Liang and G. Lee, to be published.

NCEER-91-0005 "3D-BASIS - Nonlinear Dynamic Analysis of Three Dimensional Base Isolated Structures: Part II," by S. Nagarajaiah, A.M. Reinhorn and M.C. Constantinou, 2/28/91.

NCEER-91-0006 "A Multidimensional Hysteretic Model for Plasticity Deforming Metals in Energy Absorbing Devices," by E.J. Graesser and F.A. Cozzarelli, 4/9/91, to be published.

NCEER-91-0007 "A Framework for Customizable Knowledge-Based Expert Systems with an Application to a KBES for Evaluating the Seismic Resistance of Existing Buildings," by E.G. Ibarra-Anaya and S.J. Fenves, 4/9/91.