1

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Services des thèses canadiennes

Ottawa, Canada
K1A 0N4

CANADIAN THESES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

**THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED**

THÈSES CANADIENNES

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

**LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE**

National Library
of Canada

Bibliothèque nationale
du Canada

0-315-22200-X

Canadian Theses
on Microfiche Service

Services des thèses canadiennes
sur microfiches

Ottawa, Canada
K1A 0N4

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

• Please print or type — Ecrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

STEVEN MICHAEL CORNICK

Date of Birth — Date de naissance

15 January, 1961

Country of Birth — Lieu de naissance

ENGLAND

Permanent Address — Résidence fixe

1025 RICHMOND RD #APT 2208
OTTAWA, ONTARIO, CANADA, K2B-8G8

Title of Thesis — Titre de la thèse

MASON: A RULE BASED DAMAGE
ASSESSMENT SYSTEM FOR MASONRY
WALLS.

University — Université

CARLETON UNIVERSITY

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

MASTER OF ENGINEERING

Year this degree conferred — Année d'obtention de ce grade

FALL 1985

Name of Supervisor — Nom du directeur de thèse

NEAL HOTZ

Permission is hereby granted to the NATIONAL LIBRARY OF
CANADA to microfilm this thesis and to lend or sell copies of
the film

The author reserves other publication rights and neither the
thesis nor extensive extracts from it may be printed or other-
wise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈ-
QUE NATIONALE DU CANADA de microfilmer cette thèse et de
prêter ou de vendre des exemplaires du film

L'auteur se réserve les autres droits de publication, ni la thèse
ni de longs extraits de celle-ci ne doivent être imprimés ou
autrement reproduits sans l'autorisation écrite de l'auteur

Date

29 AUGUST, 1985

Signature

Steven Cornick

MASON: A RULE BASED DAMAGE ASSESSMENT SYSTEM FOR MASONRY WALLS

by

Steven M. Cornick, B. Eng. (McGill)

A thesis submitted to the Faculty of Graduate Studies
and Research in partial fulfillment of the requirements
for the degree of

MASTER OF ENGINEERING

Faculty of Engineering

Carleton University

Ottawa, Ontario

August 1985

The undersigned recommend to the Faculty of Graduate Studies acceptance of the thesis:

**"MASON: A Rule Based Damage Assessment
System for Masonry Walls"**

submitted by Steven M. Cornick, B. Eng., in the partial fulfillment of the requirement for the degree of

MASTER OF ENGINEERING

Thesis Supervisor

**Chairman,
Department of Civil
Engineering**

Carleton University

August 1985.

Abstract

The objectives of this research were threefold. The first phase of the work was to investigate the applicability of Artificial Intelligence tools and techniques to Civil Engineering problems. Several standard problem solving strategies were investigated with the objective of creating a body of knowledge that could be used for future experimentation. The second involved the implementation of some tools that could serve as a starting point for further research. A fairly standard problem solving strategy — a backward chaining inference engine — was developed for engineering applications. A framework system for the development of engineering expert systems called EMASON was implemented. EMASON differed from currently available systems in its ability to handle moderately complex numerical calculations and its ability to allow the user to modify the system's current knowledge during execution. A rudimentary frame based representation language was also implemented and investigated for its applicability. The third objective was to demonstrate the application of the EMASON system to a Civil Engineering problem. The problem of diagnosing failures in masonry walls was chosen to best demonstrate how such Artificial Intelligence techniques could be successfully applied in the engineering domain. MASON, a system for the rule based damage assessment of masonry walls was developed using the EMASON framework. Several sample interactions with the MASON system were conducted in order to demonstrate the system's use.

Keywords: Civil Engineering, Masonry, Expert Systems, Artificial Intelligence

Acknowledgements

The author wishes to express his sincere thanks to;

Dr. N. Holtz for his unending support and seemingly infinite patience in answering the author's numerous questions.

The members of the CASCADE research group for their development and maintenance of the APOLLO computer network.

His parents for their unwavering support during the difficult periods and for providing the author with the inspiration to complete this work.

His colleagues for their timely suggestions and recommendations.

This document was entirely produced via electronic means, using a variety of software packages. At no time did a pencil (or pen) ever touch a piece of paper during its production.

Table of Contents

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	viii
1 Introduction	1
1.1 Objectives	1
1.2 Organization	2
2 Previous Research	3
2.1 A Review of Some Prominent KBES	3
2.2 Review of Prominent KBES Frameworks and Language Tools	5
2.3 Engineering Knowledge Based Expert Systems	7
2.4 Masonry Construction	10
3 Introduction to Knowledge Based Expert Systems	13
3.1 What are Knowledge Based Systems?	13
3.2 Problem Solving Strategies	16
3.2.1 Forward Chaining	16
3.2.2 Backward Chaining	18
3.2.3 Means-Ends Analysis	21
3.2.4 Problem Reduction	21
3.2.5 Plan Generate and Test	22
3.2.6 Agenda Control	23
3.2.7 Backtracking	23
3.2.8 Hierarchical Planning and Least Commitment Principle	24
3.2.9 Constraint Manipulation	25
3.3 The Knowledge Base	25
3.3.1 A Civil Engineering Example	26
3.3.2 Production Rules	29
3.3.3 Logic Programming	33
3.3.4 Frame Based Representations	37
3.4 Summary	41
4 An Overview of EMASON	42
4.1 The Choice of Language	42
4.2 The Choice of Knowledge Representation	43
4.3 A High Level Overview of EMASON	43
4.3.1 The Choice of the Inference Mechanism	44

4.3.1	The Choice of the Inference Mechanism	44
4.3.2	The Inference Engine	44
4.3.3	Requirements for an Engineering Rule Base	48
4.3.4	Form of the Production Rules	50
4.3.5	The Explanation Module	52
4.3.6	Knowledge Acquisition Tools	53
4.3.7	Natural Language Interfaces	54
4.3.8	Certainty Factors	54
4.3.9	Implementation Issues	56
4.4	Summary	57
5	The MASON Knowledge Base	58
5.1	Implementation of the Knowledge Base	58
5.2	Organization of the Knowledge Base	60
5.3	Special Rules	60
5.4	General Rules	62
5.5	Property Rules	64
5.6	Masonry Rules	66
5.7	Failure Rules	68
5.7.1	Failure Types 1 and 2	69
5.7.2	Failure Type 3	69
5.7.3	Failure Type 4	74
5.7.4	Failure Type 5	77
5.7.5	Failure Type 6	80
5.8	Summary	80
6	Sample Interactions with MASON	82
6.1	Introduction	82
6.2	Sample Session One	82
6.3	Sample Session Two	100
6.4	Sample Session Three	115
6.5	Summary	130
7	Conclusions and Recommendations	131
7.1	Introduction	131
7.2	Conclusions	131
7.3	Limitations of EMASON/MASON	134
7.4	Recommendations	135
Appendices		
A	Production Rules for Slab Design	137
B	PROLOG Source Code for Slab Design	140
C	Frame Based Source Code for Slab Design	142

D A Listing of Rules in MASON	146
D.1 Special Rules	146
D.2 General Rules	146
D.3 Property Rules	149
D.4 Masonry Rules	154
D.5 Failure Type 1 Rules	159
D.6 Failure Type 2 Rules	164
D.7 Failure Type 3 Rules	166
D.8 Failure Type 4 Rules	169
D.9 Failure Type 5 Rules	174
Glossary	179
References	182

List of Figures

3-1 A Conventional Program.	14
3-2 A Data Base Management System.	15
3-3 A Knowledge Based Expert System.	16
3-4 An Appropriate Domain for Forward Chaining.	17
3-5 An Appropriate Domain for Backward Chaining.	19
3-6 An AND-OR Graph of the sample problem.	22
3-7 Edge Support Conditions.	27
3-8 General Structure of a Frame.	38
5-1 Organization of the MASON Knowledge Base.	61
5-2 Failure Types 1 and 2.	70
5-3 Failure Type 3.	71
5-4 Stress Distributions.	72
5-5 Failure Type 4.	75
5-6 Failure Type 5.	78
6-1 Case for sample session one.	83
6-2 Case for sample session two.	101
6-3 Case for sample session three.	116

Chapter 1

Introduction

In view of the key role of design in professional activity, it is almost ironic that in this century the natural sciences have almost driven the sciences of the artificial from professional school curricula. Engineering schools have become schools of physics and mathematics; medical schools have become schools of biological science; business schools have become schools of finite mathematics. The use of adjectives like "applied" conceals, but does not change, the fact.

—H. A. Simon

The Sciences of the Artificial

Increasingly engineers are attempting to develop computer programs that solve problems in areas that do not have explicit algorithmic solutions. Two decades of numerical experience has produced a wealth of expertise in the area of applications programming and has produced a marked increase in the computational productivity of engineers. Recently the focus of attention has shifted towards the initial and final stages of problem solving and design and it has been found that conventional programming techniques are inadequate to provide effective applications programs in these areas. In response to the need for intelligent computer aided design tools the focus of current research has shifted towards the application of Artificial Intelligence (AI) tools and techniques.

1.1 Objectives

The thrust of this research was threefold. The first objective was the investigation of the applicability of some AI tools and techniques to Civil Engineering problems. This phase involved the implementation and testing of various standard problem solving strategies. The goal was to create a body of knowledge and expertise in the area of Knowledge Based Expert Systems (KBES) that can be used for further research in various areas of Civil Engineering.

The second objective was closely related to the first. Briefly stated, it was the development of a set of tools that could serve as a starting point for further research in

the applications of AI techniques. The tools developed included a Rule Based Expert System (ES) framework, a frame based representation language, and the beginning of an object oriented constraint manipulation language based on the work of Sussman and Steele [70]. All these tools were implemented in Portable Standard Lisp (PSL).

The third objective was to demonstrate the application of the ES framework by applying it to a Civil Engineering problem. During this phase it was important to limit the scope of the problem, and that could be best done by choosing a domain that was not too broad. It was decided that the field of masonry failures was sufficiently narrow, yet deep enough to serve as an adequate starting point. This is because current building practices have placed the emphasis on lightweight masonry construction and despite the state of the art, failures of varying magnitudes occur frequently. The problem of determining the possible causes of failure in a masonry walls requires considerable judgment and expertise. The MASON system, a KBES that performs rule based damage assessment of masonry walls, was implemented using the EMASON ES framework.

In the process of implementing the MASON system it was necessary to begin the development of a well formalized knowledge base concerning masonry failures. While it is hoped that this will stimulate further research in the behavior of masonry structures, the long term benefits of such a system will lie more in its educational and predictive usage (to prevent failures at the design stage) and in the development of a body knowledge concerning the application of AI techniques to Civil Engineering problems.

1.2 Organization

The remainder of this thesis is divided into six chapters. Chapter 2 reviews some important Expert Systems. The systems reviewed include those developed in the domain of Civil Engineering. The chapter also touches on some relevant aspects concerning masonry walls. Chapter 3 is an introduction to Knowledge Base Expert Systems and Artificial Intelligence tools and techniques. Chapter 4 describes the implementation of the EMASON Expert System framework. The implementation of the MASON system is discussed in Chapter 5. Chapter 6 contains some sample interactions with the MASON and EMASON systems. The conclusions of this research are presented in Chapter 7.

Chapter 2

Previous Research

The popular view that scientists proceed inexorably from well established-fact to well-established fact, never being influenced by any unproven conjecture, is quite mistaken. Provided it is made clear which are proven facts and which are conjectures, no harm can result. Conjectures are of great importance since they suggest useful lines of research.

—A. M. Turing

Computing Machinery and Intelligence

This chapter is a review of some important prominent and historic knowledge based expert systems.¹ Presently there are hundreds of KBES in existence, and the numbers are increasing rapidly — it would be impossible to review them all. A knowledge based expert system is a computer program that achieves a high level of performance in a domain that for human beings would require years of training. Frameworks and tools are programming systems and languages that simplify the work of building expert systems and acquiring knowledge. Also included is a section briefly reviewing problems of masonry construction.

2.1 A Review of Some Prominent KBES

Five of the most commonly cited KBES are reviewed in this section. They represent a variety of domains of expertise and approaches to knowledge representation and problem solving strategies.

DENDRAL [8] was the first practically successful KBES and dates from 1965. It was developed at Stanford University and is used for the identification of organic compounds by analysis of mass spectrographs. The problem here is that a mass spectrograph can be computed from an organic formula, but not vice-versa. The vast number of possible organic compounds rules out simple trial and error (generate and test) methods of solutions. DENDRAL contains roughly four hundred rules and is claimed to rival human experts in the field. The system uses rules to constrain the candidate compounds and to produce a list of possible compounds, based on a comparison of predicted spectrograph with the

¹ Certain words and phrases whose meanings may not be clear are introduced here. They are distinguished by a boldface typeface, and brief definitions may be found in the Glossary.

actual. Candidates are ranked according to the best first. DENDRAL has discovered some previously undiscovered rules and rediscovered some old rules using the META-DENDRAL package.

MYCIN [18], [57] is perhaps the most commonly referred to and copied KBES to date. It was developed at Stanford University from 1975 onwards. MYCIN is used as a consultant and, based on an interaction with a physician, diagnoses blood and meningitis disorders and recommends an appropriate drug treatment. It contains approximately four hundred rules. MYCIN is a rule based backward chaining system that uses inexact reasoning to verify hypotheses. The use of certainty factors provides the user with a measure of belief for the hypothesis considered by the system. MYCIN is written in LISP.

TEIRESIAS [19] is a knowledge acquisition program intended for use in close conjunction with MYCIN. It handles all the communications with the user and MYCIN and serves as an explanation facility and as a rather constrained natural language interface. TEIRESIAS can also be used as an assistant in the task of constructing large complex KBES. In fact TEIRESIAS is a KBES whose domain is MYCIN.

PROSPECTOR [27], developed at the Stanford Research Institute International (SRI International), is a system that aids geologists in evaluating mineral sites for evidence of potential deposits. The user provides PROSPECTOR with a list of rocks and minerals observed and other information, in a constrained form of English. The system sets up a dialogue with the user and requests information based on the hypothesis the system currently believes. PROSPECTOR is capable of explaining and rationalizing about deductions that it makes. The system is a combination of assertions and inferential rules. Estimates of the measure of belief of a hypothesis is provided on a scale between -5 and 5, -5 being strong evidence against and 5 being strong evidence for a given hypothesis. PROSPECTOR is a commercial KBES and as of 1979 consultation rates charged to clients were ten dollars per hour. This fee is quite low when the potential benefits of just one major deposit is considered.

R1 [40] (now called EXSEL) is a system developed at Carnegie-Mellon University (CMU). R1 configures VAX-11/780 computer systems from customer orders. If the orders are incomplete then R1 completes the orders according to the users requirements. The system is a rule based system using a modified forward chaining problem solving strategy.

Presently R1 contains approximately ten thousand rules and is used extensively by Digital Equipment Corporation.

MOLGEN [38] developed at the Stanford Heuristic Programming Project (HPP) provides intelligent advice to molecular geneticists on the planning of experiments involving the manipulation of DNA. MOLGEN takes a different approach to problem solving than the previous KBES reviewed. The system is based on the proposition that successful problem solving in a complicated domain needs the use of large amounts of domain specific knowledge. MOLGEN uses a *schema* or *frame based* language as opposed to the rule base based systems discussed above. Control is maintained using a *blackboard* approach that is very effective in ill structured and complex domains. The system contains *procedural attachments*, in the form of rules, if specific criteria are met. MOLGEN performs analysis, classification, design, and planning tasks.

2.2 Review of Prominent KBES Frameworks and Language Tools

As with KBES there now exists a multitude of KBES frameworks and specialized languages that have been developed for the efficient implementation of such systems. In this section only the more commonly known tools will be reviewed.

EMYCIN (Essential MYCIN) [72] is MYCIN emptied of its domain knowledge. It is a programming system for writing knowledge based consultation systems using the production rule format. EMYCIN offers advice to the user on some problem within the domain of its expertise. The system asks the user questions during the course of the consultation in a constrained form of English. Inexact reasoning in KBES constructed using EMYCIN is provided through the use of certainty factors. Currently users of EMYCIN must have an extensive knowledge of LISP. Not much is known about the limits on the fields of applications for EMYCIN. EMYCIN has spawned several KBES, among them PUFF, a system that diagnoses pulmonary disorders, ONCOCIN a consulting system for chemotherapy, HEADMED, a system that diagnoses a range of psychiatric disorders and recommends drug treatments, and SACON and SESCON for stress analysis—two systems that will be discussed below.

KAS (Knowledge Acquisition System) [51] was developed at Stanford. It was written in LISP. The KAS framework depends on a combination of rules and semantic nets. KAS is essentially PROSPECTOR emptied of all its domain knowledge and has all the same features. HYDRO, a system discussed below, was implemented using KAS.

AGE (Attempt to Generalize) [48] is another system developed at Stanford. AGE was written in LISP and was essentially a combination of the KAS and EMYCIN frameworks. The user can use either frames or rules or both. AGE provides guidance to the user on building KBES and tools for doing so. AGE was intended to make explicit the art of knowledge engineering and had two main goals. The first involved the isolation of techniques used in KBES while the second involved the building of an intelligent system guide to using these techniques. AGE contains facilities for implementing rule interpreters and control strategies, explanation modules, traces, and certainty factors. AGE uses the blackboard approach as a medium for the control strategies.

EXPERT [73] was a system developed at Rutgers University to aid in designing and building consultation models. It was developed as a generic system and was not tied to any specific application. EXPERT uses a relatively simple knowledge representation and inference procedure. The inference procedure uses a partially ordered conceptual network, a special type of semantic net. Knowledge is represented as production rules. EXPERT is written in FORTRAN, and is the only known example of a FORTRAN based system.

Currently there are many specialized languages that have been developed for the implementation of KBES, such as ROSIE[22], LOOPS[6], SRL[76], PSRL[53], and OPS5[26]. There are also a few general purpose languages that are useful for the development of KBES—namely LISP and PROLOG. These and OPS5 will be discussed later. SRL (Schema Representation Language) was developed at CMU and is a frame based representation language written in LISP. Frame based representations are discussed in Section 3.3.4. LOOPS provides a framework for constructing KBES, it combines procedure oriented, object oriented, and access oriented programming paradigms. ROSIE is a general purpose programming language, implemented in LISP that takes stylized English input to form assertions. Knowledge is represented in the form of rules and facts.

2.3 Engineering Knowledge Based Expert Systems

This section will review a few KBES that are in the engineering domain. DART [5] is a KBES developed using the EMYCIN framework. DART contains about 190 rules and is used to diagnose hardware faults in IBM370 computer systems. It was developed jointly at Stanford HPP and IBM. SU/X [49] is a system developed at SRI International that forms and updates hypotheses about the location, velocity, and origin of objects from primary signal data. SU/X was implemented in the context of military signal-understanding application. The system is rule based. VLSI [7], developed at Stanford, assists in the design of VLSI (Very Large Scale Integration) circuits.

Over the past several years several civil engineering expert systems have been developed. SACON (Structural Analysis CONSULTANT)[4] is KBES that advises engineers in the use of a large complex finite element applications program, namely MARC. SACON was developed as an intelligent consultant to the user as it was estimated that it takes a year to become proficient in the use of MARC. SACON considers 38 analysis cases, among them linear analysis, nonlinear geometric crack growth, and material instability. The structures analysed so far using the SACON-MARC combination range from aircraft wings to rocket motor casings to bridges and buildings. The system infers the critical material stresses and deflection behaviors to determine the analysis options. Structures are decomposed into sub-structures having individual geometries and material properties. SACON was developed at Stanford HPP using the EMYCIN framework and contains approximately 170 rules. The inference mechanism is a rule based backward chaining engine. Typical consultations take about a half an hour at an interactive terminal. A typical rule is shown below.

Rule: 050

IF

- 1) The material composing the sub-structure is one of the metals. and
- 2) The analysis error (in percent) that is tolerable is between 5 and 30. and
- 3) The non-dimensional stress of the sub-structure is greater than 0.9. and
- 4) The number of cycles the loading is to be applied is between 1000 and 10000.

THEN

It is definite (1.0) that fatigue is one of the stress behavior phenomena in the sub-structure.

SESCON (SESAM-69 CONSULTANT) [25] is a prototype front end for the SESAM-69 structural analysis package and is similar to SACON. It was also developed using the EMYCIN framework.

SPECON (SPECification CONSULTANT) [62] was developed at CMU as a class project. The system checks data against design specifications. It was intended to aid the engineer in checking the specifications of structural steel elements for conformance with AISC steel specifications. The system, a rule based system that performs both forward and backward chaining, was written in LISP. The system allows the user to modify the parameters of a design until it conforms to the constraints imposed on it. The syntax of the provisions have the following format.

```
< Specification Provision >
  IF
    <condition> of <applicability> is met.
  THEN
    <object> shall have <performance>.
```

SPERIL-I [32],[33] (Structural PERIL) is a system developed at Purdue University. SPERIL provides rule based damage assessment of existing structures due to earthquakes. The system was developed because state of the art damage assessment relies on a very small number of experienced engineers. It was deemed beneficial to record their knowledge for the benefit of other less experienced engineers so that training periods could be reduced. The system uses inexact reasoning by jointly applying fuzzy set theory and certainty factors. The approach to the problem was to view damage assessment as a problem of pattern recognition. Damage is estimated on scale of 0 to 10, 0 being no damage and 10 being total collapse. SPERIL relies on field observations, accelerometer readings and the results of non destructive load testing before and after an earthquake. Example of some typical rules are given below.

```
Rule:      201
           IF
             1) Diagnoses of stiffness is severe. and
             2) Material is reinforced concrete.
           THEN
             There is considerable indication (0.6) that damage
               of a global nature is severe.
```

Rule: 207

IF

- 1) Diagnoses of global nature of structural walls from field inspection is severe. and
- 2) Material is reinforced concrete.

THEN

There is considerable indication (0.5) that damage of a global nature is severe.

DESTINY [64] is a system currently under development at CMU. DESTINY is an integrated structural design system. The system is blackboard based—Knowledge Modules (KM) or knowledge specialists communicate via the blackboard. KM can contain surface or heuristic knowledge or they can contain deep knowledge, usually algorithmic processes. DESTINY contains four levels of KM. The first level is the strategy level. The KM at this level analyse the current context to determine the course of the next actions. The second level, or activation KM, know when to use the appropriate specialist KM. The specialist KM, at the third level, are in themselves KBES or applications programs that specialize in one particular area of structural design. In DESTINY the specialist KM contain such programs as CONMAN [64] and ALL-RISE.² CONMAN is a consultant for modeling and analysis whose purpose is to expand the information provided by ALL-RISE into a form suitable for SAP-IV, to perform the analysis, and to condense the results of the analysis to be sent back to the blackboard. The fourth level of KM is the resource level. At this level the modules contain the knowledge required for structural design such as building codes, design specifications and catalogs. Examples of resource KM are SAPIV (Structural Analysis Program IV), developed at Berkeley and INGRES, a relational data base. The blackboard model has been implemented in SRL and the inference mechanism controlling the activities on the blackboard is an agenda controller. DESTINY is currently under development. ALL-RISE is being completed and a general control strategy has been successfully implemented.

HI-RISE [36] is a system developed at CMU that serves as a consultant in the preliminary design of high rise structures. HI-RISE is implemented in PSRL, a hybrid language combining frame based and production rule knowledge representations. The scope of HI-RISE is presently limited to the design of rectangular commercial and residential high rise

² ALL-RISE is a proposed extension of the HI-RISE system to include low and medium rise buildings.

buildings. The system proposes structural schemes for gravity and lateral load resisting systems. Potential designs are constrained by esthetic, spatial, equilibrium, and economic factors. HI-RISE assumes initially that all possible structural systems are valid and begins to eliminate infeasible solutions. The ranking of solutions is based upon the accumulation of penalty factors. Schemes are penalized for such things as their relative cost or whether or not they cause numerous blockages in the vertical plane.

DESIGNER [37] is a KBES used in the design of ships. It handles the numerical modeling at the preliminary design stage. DESIGNER has the ability to explain how particular results were achieved and the ability to save designs in a catalogue.

HYDRO [28] is a KBES developed from the PROSPECTOR-KAS environment. It was intended to aid hydrologists in the selection of numerical input parameters for the Hydorcomp HSPF simulation program. The system incorporates judgemental expertise about the calculation of the parameters. Like PROSPECTOR, HYDRO utilizes an inference net combining assertions and rules.

It has been suggested [62] that the field of geotechnical engineering is perhaps best suited to KBES and potential development areas have been outlined. These areas include soil mechanics and foundation engineering, seismic risk analysis and seismic signal analysis. KBES in the geotechnical field could perform such things as site identification and the design of exploration programs. A more elaborate development of these areas, specifically in the field of ~~Landslide~~ Engineering was presented by Stanford [67].

An excellent bibliography on Knowledge Based Expert Systems in engineering is provided by Sriram [63].

2.4 Masonry Construction

Masonry is one of the oldest engineering materials. The use of plain masonry can be traced back to times before the early Egyptians. Until recently it has also been the least understood in terms of its structural behavior. In the past masonry structures were massive buildings whose weight tended to resist overturning—this placed a practical limitation on the height of such structures. The modern emphasis on the more efficient use of land requires the design of taller (and lighter weight) buildings, which in turn makes an understanding

of material behavior more important. This section is intended to provide a broad overview of the field of modern engineered masonry.

Many researchers have concentrated their efforts on the behavior of masonry and its interaction with supporting structures. Carter and Stafford Smith [15] [65] conducted a series of experiments to determine the behavior of infill panels in multi-storey frames. Sahlin [54] and Soame [60] discussed the interaction of brick masonry walls with concrete slabs. Holmes [30] investigated the contribution of concrete and brickwork infills to the stiffness of steel frames and suggested a simple procedure for estimating the ultimate capacity and side sway deflection of such frames. Davies and Ahmed [17] conducted an extensive analysis of wall beam interactions. The results of this work was used as the basis for a failure mechanism in the MASON system. Much research was also devoted determining the properties and behavior of the material used in masonry construction. Results of this type of research can be found in the variety of handbooks available, such as [50].

The result of many years of work has been the introduction of several building codes and specifications that are intended to aid the engineer in designing efficient masonry structures that are not prone to failure. Documents such as, CSA S304-M78 [13] and CSA A371-M1980 [12] rely heavily on previous experimental and theoretical work. Progress is continuing with the effort to introduce a limit states design code for masonry. V. C. Fenton [24] investigated a number of methods that have been proposed for the prediction of the ultimate strength of masonry. G. A. Fenton [23] derived from available experimental evidence a means to predict stresses in a veneer of a highrise building due to differential movement. A computer program, VENEER, was developed on the basis of theory presented. VENEER calculates the movements and stresses in a building at up to eight specific times during the structures lifetime. The program handles veneers for buildings with reinforced concrete, steel or masonry frames.

One of the first textbooks to treat masonry as a structural material was written by Salhin [55]. He has applied general theories of structural analysis to the design and strength prediction of masonry under nominal loading conditions and the material properities. One chapter has been devoted to cracking and contains a survey of cracking in two to nine storey apartment houses. Schnieder and Dicky [56] is representative of modern masonry texts, concentrating on the overall design of masonry. The book by Smith [59] is representative of

the practice oriented type text. Much of the text is devoted to masonry material properties and construction practices. Pages 103 to 109 deal with masonry failure modes.

Work has been done concerning cracking in masonry. A report by the National Research Council of Canada [45] addresses the issue of cracks, movements and joints in buildings. Although it does not deal with masonry exclusively, parts of the document are devoted to the problems and practices of masonry construction. A more comprehensive survey is provided by Sorenson [61]. The work by Bulger [9] examines the causes of cracking in several existing structures. The notes for a two-day course by Drysdale and Suter [21] provide an excellent overview of modern engineered masonry. The notes are based on current codes and specifications, research papers, and case studies of existing structures. The majority of the information obtained for the MASON Knowledge Base was derived from this source.

Chapter 3

Introduction to Knowledge Based Expert Systems

The original question, "Can machines think?" I believe to be too meaningless to deserve discussion. Nevertheless I believe that at the end of the century the use of words and general educated opinion will have altered so much that one be able to speak of machines thinking without expecting to be contradicted. I believe further that no useful purpose is served by concealing these beliefs.

—A. M. Turing
Computing Machinery and Intelligence

The field of expert systems involves the study of methods and techniques for constructing systems with specialised problem solving expertise. Expert Systems incorporate knowledge, expertise, and judgemental knowledge about a particular domain, in contrast to most current applications that implement well defined numerical or semi-numerical algorithms. In the context of this thesis, knowledge is loosely defined to be a collection of facts, rules that can deduce new facts, heuristics, and procedures that can manipulate these.

3.1 What are Knowledge Based Systems?

There are several reasons for focusing on knowledge rather than than formal reasoning (i.e formal logic). First, most interesting real world problems do not have known algorithmic solutions. The second reason is pragmatic — human experts achieve high performance in problem solving because they use knowledge. A third reason is that knowledge is a valuable commodity, and transferring knowledge from an expert to a novice takes a long time. Acquiring knowledge for a KBES has the consequence of reducing the cost of knowledge reproduction, transference, and exploitation.

A KBES differs in several important respects from a system built using conventional programming techniques. Conventional programming is based on the **Power Paradigm**. Programs of this type rely on general inference methods, general search algorithms, and numerical evaluation all applied within a well structured and well understood domain. The knowledge representation in the Power Paradigm is procedural. Knowledge about the problem and its solution is incorporated statically into the applications programs, thus the

task knowledge and control knowledge are combined into the system architecture, as shown in Figure 3.1. By incorporating all knowledge in the 'source code' conventional programs tend to be relatively static, difficult to modify and unable to handle unforeseen variations. Programs of this type have been, however, very successfully applied in various disciplines of engineering. Commercial finite element programs, such as NASTRAN and SAPIV provide an excellent example of conventional data processing techniques in the Power Paradigm.

All knowledge is statically encoded in the program

Figure 3-1: A Conventional Program.

Data base systems are a way to extend conventional programming techniques. A data base system is a computer based record keeping system, whose overall purpose is the recording, maintenance, and reformatting of data. Data base systems are integrated (distinct data files are amalgamated with the redundancies removed) and shared (several different users have access to the data). A Data Base Management System (DBMS) controls access and maintains the data base. The users are generally applications programs of the conventional type. Data base systems provide for independence of data from application thus allowing for centralized control of the data. Some knowledge about the domain can be incorporated in the DBMS or the data structures themselves, allowing for consistency and integrity management. Data base systems are becoming increasingly important in engineering design, as the cost of storing, managing and interpreting data increases. They are a collection of conventional programs, based on the Power Paradigm, accessing data through a centralized controller. A typical data base system is shown in Figure 3.2.

Generally, artificial intelligence (AI) applications differ from conventional data processing applications. Most AI applications use symbolic representation, symbolic inferences,

Some knowledge can be incorporated into the Data Base system and data representation.

Figure 3-2: A Data Base Management System.

and heuristic searching (examples of heuristic searches strategies, are the A* and AO* algorithms [52]). A KBES is based on the Knowledge Paradigm. Systems that use the Knowledge Paradigm typically have rule based heuristics, domain knowledge-focused searching methods, and symbolic manipulation and evaluation, making them well suited to ill structured domains. The use of domain specific problem solving strategies distinguishes a KBES from a typical AI application. A KBES uses self reasoning, not employed in most AI applications, to explain its inferences.

In a KBES the knowledge about the domain is separated from the control structure, or inference mechanism. The separation of task knowledge and data from control knowledge increases the flexibility and modularity of the system. The separation also allows for the use of control knowledge to explain the behavior system, a feature not possible with conventional programming techniques. A typical KBES has three major components: the knowledge base, the global and dynamic data base, also known as the context, and the control structure or inference engine. (The inference engine and knowledge base will be discussed in Sections 3.2 and 3.3.) The context is a representation of the current state of the problem at hand. It is a collected series of facts and symbols, consisting of all initial and generated

information. The architecture of a KBES is dependent on the inference engine, the domain, and the representation of knowledge in the knowledge base. A schematic of a KBES, though not typical, is shown in Figure 3.3.

Domain Knowledge is separate from Control Knowledge.

Figure 3-3: A Knowledge Based Expert System.

3.2 Problem Solving Strategies

In this section several problem solving strategies will be examined. These strategies are used in some current implementations of knowledge based expert systems. Strong problem solving strategies employ domain specific knowledge to achieve goals. Problem solving can be viewed as the search for a solution through a state space. The search is performed by applying a set of operators, where an operator transforms a state into another legal state. The search proceeds from the initial state of the problem to intermediate states and finally to a goal state, when the problem is solved. A more detailed discussion of state space programming can be found on page 41 of the text by Rich [52].

3.2.1 Forward Chaining

Forward chaining is also called bottom-up, data-driven or antecedent-driven chaining. A forward chaining inference engine produces new facts or decisions by trying to affirm the situation parts of inferential rules with the conditions currently believed. The new affirmed

propositions change the current set of beliefs and the procedure is repeated until a solution is found or nothing new can be deduced. The forward chaining strategy is appropriate when there are many initial states and relatively few goal states, as shown in Figure 3.4.

Figure 3-4: An Appropriate Domain for Forward Chaining.

To demonstrate how the forward chaining strategy works consider the following set of inferential rules and desired initial and goal states;

RULES:

- 1) $B \wedge D \wedge E \Rightarrow F$
- 2) $D \wedge G \Rightarrow A$
- 3) $C \wedge F \Rightarrow A$
- 4) $C \Rightarrow D$
- 5) $D \Rightarrow E$
- 6) $A \Rightarrow H$

INITIAL:

$B \wedge C$

GOAL:

H

The first rule states that if B and D and E are all known then F can be concluded. Using a forward chaining inference engine the proof is below. The rules used and the conclusions that they deduce are given on the left hand side while the current state of the system (i.e. the context) is given on the right.

Inferences:

From Rule 4) $C \Rightarrow D$ Conclude D

From Rule 5) $D \Rightarrow E$ Conclude E

From Rule 1) $B \wedge D \wedge E \Rightarrow F$ Conclude F

From Rule 3) $C \wedge F \Rightarrow A$ Conclude A

From Rule 6) $A \Rightarrow H$ Conclude H

State:

$B \wedge C$

$B \wedge C \wedge D$

$B \wedge C \wedge D \wedge E$

$B \wedge C \wedge D \wedge E \wedge F$

$B \wedge C \wedge D \wedge E \wedge F \wedge A$

$B \wedge C \wedge D \wedge E \wedge F \wedge A \wedge H$

The appropriateness of this strategy for engineering problems is dependent on the nature of the problem. Forward chaining is not appropriate for engineering design problems. Generally it is not possible to represent the goal states in an engineering design problem as a finite set of hypothesis or conclusions. Forward chaining, however, may be used for subtasks of a design problem. Where a configuration is specified or a design is substantially complete, forward chaining can be used to select an appropriate modeling option (because there are relatively few possibilities) or to verify the design checking to see if it meets specified criteria. A drawback of this problem solving strategy is that all derivable facts are derived, whether they are relevant or not.

3.2.2 Backward Chaining

A backward chaining inference engine proceeds from a goal state to an initial state. This can be viewed as a top-down approach. The solver begins with a conclusion and attempts to establish a path to an initial premise. The procedure attempts to satisfy a goal by first examining the context to see if all the immediate conditions needed to satisfy the goal have been attained. If any conditions are absent from the context these conditions are set up as subgoals and the procedure is repeated recursively. In its pure form backward chaining is a depth first search and is guaranteed to find the first solution encountered if

Figure 3-5: An Appropriate Domain for Backward Chaining.

one exists. This strategy is efficient when there are relatively few initial states as compared goal states as shown in Figure 3.5.

To demonstrate the backward chaining technique consider the same problem as above. Using a backward chaining inference engine the proof is given below:

Goal Prove H	⇒	Try Rule 6)	Prove A.
Subgoal Prove A	⇒	Try Rule 2)	Prove D ∧ G or
		Try Rule 3)	Prove C ∧ F ¹
		Try Rule 2)	Prove D ∧ G
Subgoal Prove D	⇒	Try Rule 4)	Prove C → C is known,
			Conclude D.

¹ Conflict resolution phase. Choose the first rule encountered.

Subgoal Prove G $\Rightarrow$		$\rightarrow$ G is impossible to prove Rule 2) fails.
	Try Rule 3) Prove C $\wedge$ F	
Subgoal Prove C $\Rightarrow$		$\rightarrow$ C is known.
Subgoal Prove F $\Rightarrow$	Try Rule 1) Prove B $\wedge$ D $\wedge$ E	
Subgoal Prove B $\Rightarrow$		$\rightarrow$ B is known.
Subgoal Prove D $\Rightarrow$		$\rightarrow$ D has already been established.
Subgoal Prove E $\Rightarrow$	Try Rule 5) Prove D	
Subgoal Prove D $\Rightarrow$		$\rightarrow$ D has already been established.
		$\rightarrow$ Conclude E.
		$\rightarrow$ Conclude F.
		$\rightarrow$ Conclude A.
		$\rightarrow$ Conclude H.

Backward chaining has several drawbacks, the most serious of which is the lack of backtracking. This, however, can be overcome by tracing the inference path and doing reason maintenance, at the cost of increased overhead. Another drawback, illustrated in the example, is conflict resolution, that is what should the system do if two or more rules are applicable at a given instant? The strategy used in the example is to apply the first rule encountered — this is the simplest conflict resolution strategy. There are several other strategies available. They involve the use of domain specific knowledge to resolve conflicts. Examples of these are Meta-rules (rules about how to process rules) and ranking the rules with a priority.

In its pure form backward chaining is, also not appropriate for engineering design problems, because again the goal states of design problems are not easily discretized into a finite set of hypotheses. The concept of backward chaining can be used in design problems where decomposition of the task into subtasks is needed. This strategy can also be used for diagnosing problems that have occurred in completed designs.

3.2.3 Means-Ends Analysis

The essence of means-ends analysis is the detection of differences between the current state and the desired goal state. Upon finding a difference an attempt is made to reduce the difference by applying an appropriate operator. If that operator cannot be applied to the current state then the subproblem of getting to an appropriate state where the operator can be applied is attempted. By choosing differences correctly, and defining an effective set of operators, the subproblems should be easier to solve than the original task. The procedure is applied recursively to the generated subproblems. The choice of operators and differences should be made such that the systems attention is focused on the most important aspects of the problem at hand. Means-ends analysis is a mixture of forward and backward solving strategies. To use this method on a specific problem the formulation of possible states and appropriate operators is required. It would seem that this method would be useful for engineering problems, however most real world design problems have not been sufficiently quantified and formalized to allow the type of representation required by this method.

3.2.4 Problem Reduction

Problem reduction involves reducing a problem into several subproblems. This is an appropriate strategy for most real world engineering problems. The task at hand is represented as an AND-OR graph. A node can represent a specific state of the problem at hand or a subtask to be performed. An AND node points to a series of antecedent nodes. All the precursor nodes must be satisfied to satisfy the AND node. An OR node has the same basic structure as an AND node except that only one of the antecedent nodes needs to be satisfied in order to satisfy the OR node. All of the previously discussed methods can be used to traverse an AND-OR graph. An example of such an algorithm is the AO* algorithm, given by Rich [52]. To apply this strategy an appropriate graph must be constructed for each specific problem. This can be done automatically with some specialized knowledge acquisition tools. An AND-OR graph of the set of rules described previously is shown in Figure 3.6.

AND nodes are shown with arcs between nodes.

Figure 3-6: An AND-OR Graph of the sample problem.

3.2.5 Plan Generate and Test

Generate and test is perhaps the simplest of all the problem solving strategies discussed in this section. It consists generating a proposed solution from a point in the problem space, perhaps the starting point, and testing to see if the proposed solution is actually a solution or not. If the test succeeds then the searching stops, otherwise the procedure is repeated. A systematic state space search is guaranteed to generate a solution, if one exists, although it may take a while. In order to avoid an exhaustive search, inconsistent solutions may be pruned away from earlier stages such as the planning stage. Constraints are formulated at the planning stage and the data is interpreted such that the constraints eliminate any solutions that are not viable. A drawback to the Generate and Test strategy is that if there

is no unique solution to a problem then there is obviously no absolute test for that solution. Real world problems usually have several possible solutions; thus it is necessary to apply some type of ranking procedure to estimate the appropriateness of a particular solution. This strategy seems to be appropriate for the preliminary design phase of engineering design, where several competing solutions may arise. Difficulties may arise when trying to decide on an appropriate ranking procedure.

3.2.6 Agenda Control

An agenda driven mechanism chooses the most promising tasks from an agenda list. A task is an action that when performed modifies the state of the problem at hand. An example of a task would be the pouring of foundations for a building structure. A task is justified by actions, that have previously taken place or are currently taking, that require the execution of the specific task. Once a promising task is found it is executed. The amount of resources devoted to the task depend on its importance. When a task is executed it will probably generate successor tasks—if so then those tasks are checked against the agenda list to see if they are already there. If any of the successor tasks are on the list and the justifications (i.e. precursor nodes) are the same then those tasks and their justifications are ignored.² If a successor task is not on the agenda list then it is rated and inserted into the list. The process is repeated until a solution is found or the agenda list is empty. Agenda control is useful in that some tasks may provide negative evidence as to the merits of other tasks, thereby reducing their respective rankings. This strategy allows control over the system's focus of attention. By adjusting the rankings, hence the agenda, the system's focus can be directed to a particular solution path. Agenda control seems to be appropriate for complex engineering problems that require the interaction of several different independent sources of expertise, as in the design of a large building.

3.2.7 Backtracking

Backtracking, although not a solution strategy in its own right, is an integral part of an effective solution strategy. When a problem solver finds it has taken a dead end

² In fact this represents a loop. If the task is on the agenda and the justifications are not the same then there are at least two paths to the same node. With a little extra effort the shortest path can be found and the longer path discarded.

path, it backtracks up to a point where the solution space is consistent and then proceeds along another path. In pure backtracking the solver is backed up to the starting node. Backtracking is a feature in many AI languages, such as PROLOG (PROgramming LOGic) and is described by Clocksin and Mellish [16]. For large problems pure backtracking is inefficient. Stallman and Sussman [66] have developed the concept of dependency directed backtracking. This involves formulating constraints at the outset, adding them to the data base and treating them like other information until an inconsistency is found. Instead of backing up to the starting point the system only backtracks to the decisions that led directly to the contradiction, and only these decisions are undone. The concept of dependency directed backtracking has been extended by Doyle [20] in his work on Reason Maintenance Systems.

3.2.8 Hierarchical Planning and Least Commitment Principle

Hierarchical planning involves developing a plan at successive levels. An example of hierarchical planning is perhaps more illustrative than a general description. Consider a building structure. It is composed of beams, columns, connections, floors, footings, and so on. It is impractical to solve structural engineering problems at a level of abstraction that deals with individual components, therefore higher levels of abstractions are made. At a higher level a building structure can be seen to be composed of vertical subsystems, such as frames, horizontal subsystems, such as floor systems, and foundations. At an even higher level the entire structure may be modeled according to its behavior, thus an entire building can be viewed as a simple beam or cantilever beam.

The least commitment principle is the deferring of decisions as much as possible. By relegating decisions to lower levels of abstraction solutions may be suggested, constrained or eliminated. Least commitment also tends to minimize the tasks that need to be undone if a proposed solution is found to be unsatisfactory. This method is appropriate for engineering problems, in fact most engineering problems are hierarchical or almost hierarchical in nature[58], [70].

3.2.9 Constraint Manipulation

One of the most important aspects of engineering problems is constraint manipulation. Consider, again, a large building. In such a structure there are constraints that govern the interaction between major subsystems, such as mechanical and structural systems, architectural constraints, material constraints, economic constraints, such as the cost of materials and labor, time constraints, and constraints imposed by various building codes and specifications. There are two broad categories of constraints, hard constraints (such as strength), which can not be violated without compromising the integrity of the structure, and soft constraints (such as economics), which can be violated if necessary. A solution will attempt to satisfy most or all of the imposed constraints. A simple example of constraint satisfaction and its potential uses is given by Sussman and Steel [70],[68]. They have developed a programming language based on constraints. Stefik [69] has proposed an extension to constraint satisfaction called constraint posting. Constraint posting involves three stages, constraint formulation, where new constraints are created, constraint propagation, where constraints are propagated to other parts of the problem, and constraint satisfaction, where an attempt is made to satisfy the constraints. A system that uses constraint posting and evaluates constraints symbolically and numerically was developed by Holtz [31]. Constraint manipulation is not appropriate to all engineering problems, specifically those problems that are not well formalized or problems that have informal constraints, such as individual preference. Constraint manipulation is appropriate if the subgoals of a given problem interact. An adequately designed KBES should contain some degree of constraint handling, symbolic and numeric.

3.3 The Knowledge Base

The second major component in a KBES is the knowledge base. The knowledge base is a collection of facts and rules, some heuristic, that describe a particular domain. On an abstract level, knowledge about a domain can be viewed as consisting of descriptions, relationships, and procedures. Descriptions identify and differentiate objects and classes. Relationships describe dependencies and associations between objects and classes. Procedures specify operations to perform when attempting to solve a problem in the domain.

Knowledge can be divided into two relatively broad categories; public facts and private rules. Public facts consist of theories, definitions, and facts that can usually be found in textbooks, references, codes, and specifications. Private rules are usually not published or generally available. Private rules can loosely be described as experience, rules of thumb, the ability to recognize promising approaches and the ability to deal effectively with incomplete information.

The form of the knowledge base depends on the domain and the problem that the system is intended to solve. The form also depends on the type of knowledge representation formalism used. Knowledge representation is a major area in AI research and there exist several well developed formalizations for representing knowledge. Two examples of representational techniques are semantic nets and conceptual dependencies and are described by Rich [52]. Some of the formalisms that are appropriate for engineering problems will be considered here; those examined are production rules, logic programming and frame based representations.

A Civil Engineering problem will be considered in the next section in order to demonstrate the applicability and the attributes of each of the representation schemes. The example is taken from Maher [35] and was implemented by the author. The example is useful in that the formulation of the problem embodies several of the concepts discussed in Section 3.2.

3.3.1 A Civil Engineering Example

Consider the selection of alternative floor systems in the preliminary design phase of a building. To simplify the problem somewhat the following assumptions were made;

- 1) A rectangular grid will be used to represent a typical interior bay within the building.
- 2) Only the interior bays will be considered and they are assumed to be continuous over adjacent bays.
- 3) The grid dimensions are; B the larger dimension and D , the smaller dimension.
- 4) The edge support conditions are initially unconstrained.

Using the problem reduction approach, the task of choosing a floor system is subdivided into three subproblems. Constraints formulated at the outset are used to reduce the problem

space. The information contained in the constraints was taken from Maher [35] and can also be found in Lin [34]. The first subproblem is the selection of the mode of action of the system. A floor system will behave in two-way action if the aspect ratio is near unity; those with large aspect ratios behave in one-way bending action. The second subproblem is the selection of an appropriate structural material. Four types of materials are considered; steel decking, prefabricated concrete panels, reinforced concrete, and prestressed concrete. The problem of selecting a suitable material is related to the problem of choosing an appropriate mode of behavior. For example, steel decking can only be used in one-way action. The third subproblem is the selection of suitable edge support conditions. The choice of support conditions depends on the mode of action and the type of material used. The edge support conditions are shown in Figure 3.7.

Figure 3-7: Edge Support Conditions [35].

The constraints governing the behavioral modes are given below.

```

IF
   $B/D \leq 1.5$ 
THEN
  Two-way action is possible.

IF
   $B/D > 1.2$ 
THEN
  One-way action is possible.

```

These constraints illustrate the least commitment principle in that a decision on the mode of action in the range between 1.2 and 1.5 is deferred until further options have been explored. Two types of constraints are imposed on the selection of structural materials. The first set deals with the economical spanning ranges of each material while the second set is based on the interaction of the material and the mode of behavior of the floor system. The constraints are shown as follows.

```

IF
   $Span \leq 4.0m.$ 
THEN
  Can use steel deck.

IF
   $Span \leq 5.0m.$ 
THEN
  Can use prefabricated panels.

IF
   $3.0 \leq Span \leq 8.0m.$ 
THEN
  Can use reinforced concrete.

IF
   $6.0 \leq Span \leq 13.0m.$ 
THEN
  Can use prestressed concrete.

IF
  One-way action.
THEN
  Can use steel deck, and
  Can use prefabricated panels, and
  Can use reinforced concrete, and
  Can use prestressed concrete.

IF
  Two-way action.
THEN
  Can use reinforced concrete, and
  Can use prestressed concrete.

```

The third class of constraints involves the dependence of the support conditions on the material and the mode of behavior.

```

      IF
        Can use steel deck, or
        Can use prefabricated panels.
    THEN
      Use 2-edge support.

      IF
        Can use reinforced concrete, or
        Can use prestressed concrete.
    THEN
      Use 0-edge support, or
      Use 2-edge support, or
      Use 4-edge support.

      IF
        One-way action.
    THEN
      Use 0-edge support, or
      Use 2-edge support.

      IF
        Two-way action.
    THEN
      Use 0-edge support, or
      Use 4-edge support.
  
```

Using the constraints outlined above and the subproblems described at the outset the task of choosing alternate floor systems is solved using three different representation schemes. The control strategy used in each implementation depends on the attributes of the specific formalism.

3.3.2 Production Rules

Production systems, or as they are more commonly called, rule based systems, consist of three major components. The first component is the control strategy or inference engine (control strategies were discussed in Section 3.2). The second component of a production system is a temporary data base containing information pertinent to the current problem. This is the context. The last major component is a knowledge base containing a set of productions or rules.

Rules in knowledge base consist of antecedent part containing a set of patterns or conditions. The antecedent part determines the applicability of the rule: Inferential rules

also contain a consequent part consisting of a set of actions to be taken if the conditions of the antecedent part are satisfied. Generally the inference engine uses the action part of a rule to modify the context and thus to guide the inference engine through the rule base. Production rules are largely procedural in nature in that they describe actions to be taken if certain conditions are satisfied (A declarative form of knowledge representation is examined in Section 3.3.3.). Production rules are a familiar construct in computer programming and have the following general format;

Rule:

```

      IF
        Condition(s)
    THEN
        Action(s)
  
```

Several types of knowledge can be incorporated into production rules. Mathematical expressions, calculated values, and factual information can be easily expressed, as well as heuristic rules. To illustrate the ease with which knowledge can be expressed in a production rule format consider clause 13.6 a) (i) taken from CSA S16.1-M78 [14]. The clause, expressed in the form of a production, is shown below.

Rule:

```

      IF
        Continuous lateral support not provided.
    THEN
        Clause 13.6 applies.
  
```

Rule:

```

      IF
        Clause 13.6 applies. and
         $M_u > 2M_p/3$ , and
        Member is doubly symmetric. and
        Member is class 1.
    THEN
         $M_r = 1.15\phi M_p(1 - 0.28M_p/M_u)$ 
  
```

Rule:

```

      IF
        Clause 13.6 applies. and
         $M_u > 2M_p/3$ , and
        Member is doubly symmetric. and
        Member is class 2.
    THEN
         $M_r = 1.15\phi M_p(1 - 0.28M_p/M_u)$ 
  
```

Rule:

IF
 $M_r > \phi M_p^o$
 THEN
 $M_r = \phi M_p$
 (From CSA S16.1-M78 Clause 13.6 a) (i)).

The formulation of the Civil Engineering problem just described is directly translatable into a set of production rules; these are presented in Appendix A. The constraints have been modified somewhat to suit the particular system implemented by the author. The form of the rules and the inference engine will be discussed in Chapter 4.

Consider the alternatives for a grid having dimensions of 10 m by 10 m. For the purposes of illustration the productions will be traced for the steel decking and prestressed concrete options. The first goal is to establish that steel decking is a valid alternative. Searching through the rule base the system finds an appropriate rule. Rule ID9 states that:

Rule: ID9
 IF
 One-way action, and
 Use 2-edge support, and
 Steel deck is valid, and
 THEN
 Use one-way steel deck.

It is necessary to establish the conditions in the antecedent part of the production before the actions can be taken, therefore the subgoal of establishing one-way action is set up. Rule ID1 states that:

Rule: ID1
 IF
 $B/D > 1.2$
 THEN
 One-way action.

The aspect ratio, in this case, is unity and the above rule is not satisfied. The system can not find another rule that will establish one-way action it therefore concludes that steel is not an appropriate material for the floor system. The next goal, to establish that prestressed concrete is a possible alternative, is set up. Rule ID3 states that:

Rule: ID3
 IF
 Two-way action, and
 Prestressed concrete is valid.
 THEN
 Use prestressed concrete slab.

By applying Rule ID2, the condition that two-way action is valid is satisfied.

Rule: ID2
 IF
 $B/D \leq 1.5$
 THEN
 Two-way action.

It is now necessary to establish that prestressed concrete is a valid material. Since this is not yet known it is set up as a subgoal. The system finds that to establish that subgoal, Rule ID8 must be satisfied.

Rule: ID8
 IF
 $6.0 \leq \text{Span} \leq 13.0\text{m}$.
 THEN
 Prestressed concrete is valid.

The value of span was determined by evaluating rule ID6.

Rule: ID6
 IF
 The current context is two-way.
 THEN
 $\text{Span} = B$

Given the values $B=10$ m and $D=10$ m all of the above rules were satisfied and the system concludes that a two-way prestressed concrete slab was a possible floor system. The sample interaction is shown as follows.³

rules>start

Please enter a value for the variable:

B

units>METERS

input>10.0

(VARIABLE > B = 10.0)

³ The user input is given in a *slanted* font.

Please enter a value for the variable:

D

units>METERS

input>10.0

(VARIABLE > D = 10.0)

(VARIABLE > SPAN = 10.0)

(POSSIBLE FLOOR SYSTEM IS: USE TWO WAY PRESTRESSED CONCRETE
SLAB SUPPORTED ON EITHER ZERO TWO OR FOUR EDGES)

(ALL DONE)

3.3.3 Logic Programming

In the previous section a procedural form of knowledge representation, the production rule, was examined. In this section a declarative representation will be examined. The logical formalism is useful in that new knowledge can be derived from old using mathematical deduction. Predicate logic permits the representation of facts as objects or classes of objects. The fact that all beams have supports can be represented by the statement; If X is a beam then X has supports, or, using predicate calculus notation:

$$\text{beam}(X) \rightarrow \text{has_supports}(X)$$

For a detailed treatment of predicate calculus see Mendelson [39]. Once a set of Well Formed Formulae are derived to represent a problem the Resolution Principle can be used to generate solutions if they exist. The Resolution Principle is a rule of inference that describes how one proposition can follow from another, providing an automatic method for proving theorems from axioms. Given the fact base A and a set of rules of the form $A \rightarrow B$, B can be proven automatically.

PROLOG is based on the idea of a theorem prover, using the Resolution and Unification Principles. The theorem prover is applied to a knowledge base of well formed statements. A PROLOG data base, in a general form, consists of a set of rules describing the relationships between objects and classes of objects. A template rule is shown below

$$R_0 \leftarrow R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge \dots \wedge R_n \quad n \geq 0$$

If $n=0$ then the rule represents a fact. Each term on the right hand side of the rule may contain constants, variables, or other terms. To establish the left hand side all the terms on the right hand side must be satisfied. Proceeding from left to right PROLOG searches through the rule until each term is satisfied or all the alternatives have been explored. The usual searching strategy is Depth First, although it can be modified to handle other search strategies such as Breadth First. PROLOG also incorporates backtracking when a path is found to be a dead end.

The Civil Engineering problem outlined above can easily be implemented in PROLOG. The version of PROLOG used is C-PROLOG. In C-PROLOG variables, such as "Action", begin with an upper case letter while constants, such as "one_way", begin with a lower case letter. Consider the following constraints.

```

IF
  One-way action.
THEN
  Use 0-edge support, or
  Use 2-edge support.

IF
  One-way action.
THEN
  Can use steel deck, and
  Can use prefabricated panels, and
  Can use reinforced concrete, and
  Can use prestressed concrete.

```

In PROLOG they can be represented as facts.

```

support_allowable(one_way, 0).
support_allowable(two_way, 2).

material_action(one_way, steel_deck).
material_action(one_way, prefab_panels).
material_action(one_way, reinf_concrete).
material_action(one_way, prestr_concrete).

```

(Read as: If one_way material_action then steel_deck)

All the constraints outlined in the problem definition can be transformed into facts of this type. It is also necessary to construct a set of rules to relate the facts to each other. Consider the following constraints:

```

IF
  B/D <= 1.5
THEN
  Two-way action

IF
  B/D > 1.2
THEN
  One-way action

```

The corresponding PROLOG facts are:

```

action_type(one_way, 1.2, 20.0).
action_type(two_way, 0.0, 1.5).

```

While the meaning of the production rule version of these constraints is clear the PROLOG facts appear to be more obscure. The fact above should be read as: "If B/D is between 1.2 and 20.0 then action type may be one way." It is then necessary to construct a rule that uses these parameters to determine the mode of behavior of a particular floor panel given the dimensions. A rule that performs this function is given below.⁴

```

slab_action(B, D, Action) :-
    Ratio is B / D;
    action_type(Action, X, Y),
    Ratio > X,
    Ratio <= Y.

```

To illustrate how this rule is applied suppose that B and D each have the value of 10 m. The action type is to be determined.

```

slab_action(10, 10, Action).

```

{Evaluating from right to left PROLOG
will perform the following steps. }

⁴ In C-PROLOG the assignment operator is 'is', stating that X is bound to A . Equal ('=') is a test for equality.

1) Ratio is $B / D = 10.0 / 10.0 = 1.0 \rightarrow \text{True.}$

{ Bind Ratio to 1.0. }

2) action_type(Action, X, Y)

{ Start at the top of the data base,
try action_type(one_way 1.2, 20.0), place
marker here. Instantiate Action to one_way,
X to 1.2 and Y to 20.0. }

3) Ratio > 1.2 $\rightarrow \text{False}$

{ Return to marker, this in effect is
backtracking, try the next fact,
action_type(two_way, 0.0 1.5). Instantiate
Action to two_way, X to 0.0 and Y to 1.5 }

4) Ratio > 0.0 $\rightarrow \text{True.}$

5) Ratio $\leq 1.5 \rightarrow \text{True.}$

{ Rule is satisfied and return the value
for Action. }

6) Action = two_way $\rightarrow \text{True.}$

The PROLOG code, also from Maher [35] and implemented by the author, is given in Appendix B. Its brevity and elegance should be compared with the implementations of the formalisms described in the preceding and following sections. A sample interaction is shown below. The example lists all the valid combinations of material types, action types, and support conditions.

?-slab_parameter(3.0, 2.0, Material, Action, Support).

Material = steel_deck
 Action = one_way
 Support = 2 ;

Material = prefab_panels
 Action = one_way
 Support = 2 ;

Material = reinf_slab
 Action = two_way
 Support = 0 ;

Material = reinf_slab
 Action = two_way
 Support = 2 ;

Material = reinf_slab
 Action = two_way
 Support = 4 ;

no

3.3.4 Frame Based Representations

Two general types of knowledge representations have been examined so far, namely procedural and declarative types. Frame based representations are a hybrid of the two. The concept of frames was developed by Minsky in a paper entitled "A Conceptual Framework for Representing Knowledge" that appears in "The Psychology of Computer Vision" [42]. Frames can be viewed as complex semantic nets that typically have a great deal of internal structure. They are a general purpose data structure in which specific domain knowledge can be embedded. Frames are generally used for specific types of problem solving tasks that require an internal structure.

A frame is a collection of slots that describe the attributes of an object or of a class of objects. Each slot may contain several items. A slot usually contains a value for a specific attribute, such as, for example, the width of a floor slab. It may also contain a default value to be used when nothing is known about that attribute. Slots may also contain

conditions that must be satisfied before the slot may be filled. The feature distinguishing frames from declarative representation schemes is that the slots may also contain procedural attachments, which are pieces of code to be executed under certain conditions. Procedural attachments generally take one of the following forms; *If-needed*, *If-added*, and *If-removed*. The *If-needed* class of procedures specify a means for obtaining a value if it is needed but currently not known. The *If-added* type functions specify any necessary actions that must be performed if a slot is filled. The *If-removed* class of functions serve as compliments to the *If-added* functions. A slot may have one or all the facets described above. The generalized structure of a frame is shown in Figure 3.8.

Figure 3-8: General Structure of a Frame.

A particularly important feature of frames is the ability of a frame to inherit information from other frames. Expressed another way it is the ability of a frame to fill the slot of another associated frame. Inheritance reduces the amount of redundant information stored in the system. Frame based systems operate by choosing a candidate frame. If any of the required slots in the candidate frame are empty the system attempts to fill them using the procedural attachments. If the slots cannot be filled the system chooses another candidate frame or suggests modifications to the problem or data that would make the candidate suitable. The frame for the Civil Engineering example described earlier is shown below, where B is a slot representing the width attribute, its value is currently unknown and the procedure *Prompt_user* in the *If-needed* facet is executed whenever the value is required.

{Panel_1

Slot	Facet	Value
B	Value	< >
	If-needed	<Prompt_user>
D	Value	< >
	If-needed	<Prompt_user>
Ratio	Value	< >
	If-needed	<Calculate_ratio>
Material	Value	< >
	If-needed	<Select_materials>
Action_type	Value	< >
	If-needed	<Select_action_type>
Edge_support	Value	< >
	If-needed	<Select_edge_supports>
Alternatives	Value	< >
	If-needed	<Select_alternatives>

The *If-needed* facet contain the procedural attachments that fill in the attributes of the floor system if they are not known. If the user requests that the system retrieve the alternative floor systems, the system will attempt to get values of the alternatives. Since the slot is empty the system will execute the *select_alternatives* procedure. The *select_alternatives* procedure, however requests that the system retrieve the value of the *Material* slot. Since that slot is also empty the *select_materials* procedure is executed. The procedural attachments will call each other as needed until the desired slot is filled in with an appropriate value. In this case the user will finally be prompted by the system for the values of *B* and *D*. When the desired slot is filled in its value is returned. The implementation of the example considered previously is given in Appendix C.⁵ A sample interaction is shown below, again showing all valid combinations of results.

⁵ The implementation of the frame based representation language, however, is not given. A concise and instructive implementation is given in Chapter 22 of Winston and Horn [75].

frames>(initialize grid_frame panel_1)

T

frames>(? panel_1 alternatives)

(PLEASE ENTER A VALUE FOR SLOT: D)

input>8.0

(PLEASE ENTER A VALUE FOR SLOT: B)

input>10.0

Material: REINF_CONCRETE
Action : ONE-WAY
Support : 0

Material: REINF_CONCRETE
Action : ONE-WAY
Support : 2

Material: PRESTR_CONCRETE
Action : ONE-WAY
Support : 0

Material: PRESTR_CONCRETE
Action : ONE-WAY
Support : 2

Material: PRESTR_CONCRETE
Action : TWO-WAY
Support : 0

Material: PRESTR_CONCRETE
Action : TWO-WAY
Support : 2

Material: PRESTR_CONCRETE
Action : TWO-WAY
Support : 4

T

3.4 Summary

After reviewing various problem solving strategies it is clear that individually, in their pure form, they are unsatisfactory to handle moderately complex real world problems. An appropriate strategy for a KBES would be a mixture of several of the outlined strategies operating at various levels of interaction. It is also clear that a problem solving strategy for a KBES will depend upon the type of problem it is intended to solve. A solution strategy for the design of large building might start out using hierarchical planning and the least commitment principle. The hierarchical planner would use backward chaining techniques to divide the problem into subproblems. Also at the start the system would formulate a set of constraints according to plans and codes. Once the problem has been reduced to manageable portions, a plan, generate and test strategy might be used to examine various structural options, using constraint handling to prune away solutions that are not viable. If a dead end is reached then the system would use dependency directed backtracking to recover. The choice of knowledge representation is dependent on the nature of problem and the domain. As demonstrated in the sample problem solved, the problem solving strategy and knowledge representation formalism used in a KBES are inseparable. It is clear that a hybrid representation scheme would be appropriate for engineering problems. Frames appear to be the most promising avenue of research combining the advantages of declarative and procedural representation. HI-RISE, an expert system for the preliminary design of high rise buildings was implemented in the PSRL programming language. PSRL is a hybrid of the frame based Schematic Representation Language (SRL) and OPS5, a production rule based language.

MASON and EMASON, the systems described in this thesis, use a backward chaining inference engine that operates on a collection of production rules.

Chapter 4

An Overview of EMASON

*Facts are simple and facts are straight
Facts are lazy and facts are late
Facts all come with points of view
Facts don't do what I want them to
Facts just twist the truth around
Facts are living turned inside out
Facts are getting the best of them
Facts are nothing on the face of things
Facts don't stain the furniture
Facts go out and slam the door
Facts are written all over your face
Facts continue to change their shape*

— David Byrne and Brian Eno
Cross-eyed and Painless

This chapter discusses the implementation of EMASON. EMASON is a domain independent Expert System framework that can be used to develop ruled based expert systems. The system contains an inference engine and a set of knowledge acquisition tools to be used to construct a knowledge base in a given domain. Existing expert system frameworks and KBES tools were not used in its implementation. In implementing the system from the ground up a body of knowledge and expertise about KBES and their potential applications to engineering problems was developed.

4.1 The Choice of Language

When constructing a KBES, the following options are available:

- 1) Use existing ES frameworks and tools (such as EMYCIN or KAS).
- 2) Construct an ES framework or tool using a specialized language (such as SRL or OPSn).
- 3) Construct the KBES using a general purpose programming language (such as LISP or PROLOG).
- 4) Construct the KBES using traditional languages such as FORTRAN.

At the beginning of the project options 1 and 2 were not attractive because of the limited number of tools that were running on the available hardware. That situation has improved somewhat and option 1 is certainly preferable, when possible — it was unfortunately not possible during the early stages of this project.

FORTTRAN, the standard engineering language, was deemed inappropriate for developing an experimental KBES. Implementing EMASON in FORTRAN would involve the difficult task of extending the language to give it the ability to manipulate symbols and lists. It would be difficult to represent knowledge in a flexible manner and extremely difficult to create the highly recursive algorithms, necessary for most problem solving strategies, in an easily understood and modifiable manner. Nevertheless it has been done; a KBES tool called EXPERT, developed at Rutgers University, was written in FORTRAN [73]. EXPERT is a rule based system for developing consultation models.

LISP (LIST Processing) has several desirable attributes that made it the language of choice for implementing EMASON. The most important feature is LISP's ability to perform symbolic manipulation. LISP is an interactive language designed for interactive programming with rapid response at a terminal. Programs and data can be displayed or altered at will and during execution. LISP is an easy language to learn and requires very little time to achieve a high level of proficiency. The availability of powerful editing and debugging tools was a further reason for choosing LISP to implement EMASON.

4.2 The Choice of Knowledge Representation

Of the three knowledge representation formalisms examined in Chapter 3, production rules were chosen as the underlying knowledge representation scheme in EMASON. Production rules are the simplest and most well understood of the formalisms. In the course of the research a rudimentary frame based representation language was implemented. Research into a hybrid system consisting of a frame based language augmented by production rules has been proposed as a continuation of this work.

4.3 A High Level Overview of EMASON

This section will present an overview of EMASON. The overview closely follows the layout of a KBES shown in Figure 3.3.

4.3.1 The Choice of the Inference Mechanism

The choice of the inference mechanism was governed by the form of the problem and the nature of domain. The problem examined was diagnostic in nature. Diagnostic systems such as MYCIN have shown that a backwards chaining mechanism can be applied successfully to problems of this type. It was also felt that it was possible to categorize the failure mechanisms into a discreet set of hypotheses. Other problem solving strategies were considered, however, the task of implementing a complex and correct KBES framework was considered to be outside the scope of the proposed research. A relatively simple and easy to implement KBES was chosen in order to obtain an understanding of the concepts and problems involved in applying KBES techniques to the Civil Engineering domain.

4.3.2 The Inference Engine

The inference engine built into EMASON was relatively simple. It was roughly divided into two classes of functions; 1) functions that manipulate the production rules and the current context and 2) procedures that control the sequence of rule firings. These functions will be examined at a high level of abstraction.

There are two major functions that manipulate the context, REMEMBER and RECALL. RECALL examines the context to see if some specific information is contained within the context.

RECALL fact :

```
<IF <fact is contained in the context>
  THEN <return fact>
  OTHERWISE <return false>>
```

REMEMBER adds a specific piece of information to the context if it is not already known.

REMEMBER fact :

```
<IF <fact is contained in the context>
  THEN <return false>
  OTHERWISE <add fact to the context>
  <return fact>>
```

All access to the global and dynamic data bases are handled through these two functions.

There are three high level functions that manipulate the production rules in the knowledge base. The low level functions that extract rules from the knowledge base and retrieve appropriate portions will not be described here. The low level functions are dependent on the implementation of the rule base whereas the high level functions operate on more abstract data types. The procedure TEST_IF_CLAUSES examines the context attempting to verify that the antecedents of a given rule are satisfied. If all the antecedents are satisfied then TEST_IF_CLAUSES returns TRUE.

```
TEST_IF_CLAUSES rule_to_test :
  <FOREACH if_clause IN rule_to_test DO
    <IF <if_clause is not satisfied>
      THEN <return false>>>
  <return true>
```

TEST_IF_CLAUSES+ is similar to TEST_IF_CLAUSE except that if any of the conditions in a rule are not met it tries to prove the unsatisfied conditions. If the unsatisfied conditions cannot be proven, FALSE is returned. This will be described in more detail later.

```
TEST_IF_CLAUSES+ rule_to_test :
  <FOREACH if_clause IN rule_to_test DO
    <IF <if_clause is not satisfied>
      AND
      <if_clause cannot be proved>>
      THEN <return false>>>
  <return true>
```

The procedure USE_THEN_PARTS checks the action parts of a rule. If a consequent has not been performed then the consequent is executed. If all the consequents have been performed then nothing is done and the rule is ignored.

```
USE_THEN_PARTS rule_to_test :
  <IF <all actions in rule_to_test have been performed>
    THEN <return false>
```



```

    OTHERWISE <FOREACH unperformed_action IN then_parts DO
        <perform unperformed_action>
    <return true>>>

```

TRY_RULE and TRY_RULE+ try a rule to determine whether or not it can be fired. If all the conditions are met then the rule is fired. A rule is said to be fired if all the conditions on the left hand side are satisfied and there is at least one unperformed action on the right hand side. Thus a rule will be fired if TEST_CLAUSES or TEST_CLAUSES+ and USE_THEN_PARTS return TRUE.

```

TRY_RULE rule_to_try :
    <IF <call TEST_IF_CLAUSES rule_to_try>
        AND
        <call USE_THEN_PARTS rule_to_try>
        THEN <return true>
    OTHERWISE <return false>>

```

TRY_RULE+ is similar to TRY_RULE except that it uses TEST_IF_CLAUSES+. The functions described above are used to fire rules and modify the current context when a rule is fired.

The following procedures control the sequence of rule firings. These functions are used to manipulate the knowledge base. There are two high level functions in EMASON, VERIFY_A_FACT and STEPFORWARD. VERIFY_A_FACT tries to verify the validity of a specific piece of information. The procedure first checks to see whether or not the required information is contained in the current context. If not, VERIFY_A_FACT makes a list of all rules that can derive the requested information. If that list is empty then there are no rules that can derive the requested information and the user is queried as to the validity or value of the information. If the list is not empty the procedure then tries to deduce the fact directly by attempting to fire all the rules in the list. If any necessary conditions are lacking in the context then VERIFY_A_FACT recursively tries to satisfy them as subgoals necessary for the establishing of the original goal (thus VERIFY_A_FACT is a backward chaining procedure).

```

VERIFY_A_FACT fact :
    <IF <fact is known>

```

```

THEN <return true>
OTHERWISE <make a list a of all rules with fact in
           their action parts>
<IF <rule list is empty>
  THEN <prompt user for fact>
  OTHERWISE <FOREACH rule IN rule list DO
    <IF <call TRY_RULE rule>
      THEN <return true>
      OTHERWISE <IF <call TRY_RULE+>
        THEN <return true>>>>>>
  <return false>

```

At this point it is appropriate to take a closer look at the TEST_IF_CLAUSES+ procedure. TEST_IF_CLAUSES+ is used to indirectly prove a fact. An expanded description is shown below.

```

TEST_IF_CLAUSES+ rule_to_test
<FOREACH if_clause IN rule_to_test DO
  <IF <if_clause is not satisfied>
    THEN
      <IF <call to VERIFY_A_FACT if_clause
        returns false>
      THEN <return false>>>
  <return true>

```

Here we see that if the if-clause is not satisfied, then the call to VERIFY_A_FACT sets up a subgoal that must be proved in an attempt to prove the original

STEPFORWARD is forward chaining mechanism that steps through the knowledge base examining each rule to see whether or not it can be fired. STEPFORWARD cycles through the knowledge base once and stops when a rule is fired. The first applicable rule encountered is fired. A forward chaining inference machine, such as DEDUCE, can be constructed by embedding the STEPFORWARD procedure in a looping function that calls STEPFORWARD until no rules can be fired. Forward chaining inference methods are discussed in Section 3.2.1.

STEPFORWARD ()

```
<FOREACH rule IN the knowledge base DO
  <IF <no more rules to try>
    THEN <return false>
  OTHERWISE <IF <call TRY_RULE rule_to_try>
    THEN <return true>
    OTHERWISE <continue>>>>
```

DEDUCE ()

```
<IF <call STEPFORWARD return false>
  THEN <return false>
  OTHERWISE <call DEDUCE>>
```

The above ten functions form the kernel of the inference engine in EMASON. The form of the inference engine is based on Chapter 18 of Winston and Horn [75] with some extensive modifications.

4.3.3 Requirements for an Engineering Rule Base

Unlike Knowledge Based Expert Systems developed in other fields, such as medicine, those in engineering must be considerably more flexible. KBES must allow the engineer to do considerable exploration of alternatives, particularly in the design phase, but also in diagnostic work.

In implementing a rule base certain other requirements pertinent to engineering problems became apparent. The major requirement of an engineering knowledge base is the ability to handle many arithmetic expressions and variables. The consequence of handling arithmetic expressions is that a logical and consistent way of treating variables must be incorporated into the system.

Arithmetic expressions are conveniently handled as evaluable LISP expressions. Numeric variables in an EMASON rule base are either calculated from productions or supplied by the user. Numeric variables are differentiated syntactically and appear only in evaluable expressions. Variables appear as an argument to the function GET_VAL, which simply

retrieves the value of the variable. If a variable is not bound (i.e. has no value) then GET_VAL calls VERIFY_A_FACT and EMASON backward chains until a value for the variable is obtained either by computation or by interactively querying the user. Non-numeric variables are syntactically differentiated by preceeding the variable with a '\$' sign, such as '\$building'. Non-numeric variables provided the system with a more specific user interface facility as they allow substitution of identifiers that are significant to the user, as in query

(IS THIS TRUE $\rightarrow$ EAST WALL IS BUILT OF CLAY BRICKS ?)

which is constructed from the antecedent:

(IS THIS TRUE $\rightarrow$ \$WALL IS BUILT OF CLAY BRICKS ?)

Variables bound during execution are passed around EMASON via a global variable called the environment. If EMASON has discovered that it is unable to establish a hypothesis then the environment accumulated to that point is discarded and certain specific variables become unbound.

The ability to handle arithmetic expressions and variables immediately brings to light another requirement. In an engineering KBES the user should be able to modify certain parameters during execution. The ability to modify parameters in EMASON was restricted to basic ingredients (i.e. parameters supplied by the user). It was decided not to allow the user to modify any intermediate values obtained from the evaluation of arithmetic expressions. Allowing changes in intermediate values would require a constraint manipulation system that propagates changes both forwards and backwards along the inference paths [68], [31]. To illustrate the difficulties involved consider a problem where the values for the yield strength of steel are constrained to 300 and 450 Mpa. When the user changes an intermediate value the result of that change must be propagated backwards. It is extremely likely that a change in an intermediate value will violate the constraints imposed on the basic ingredients.

Changing a basic ingredient may introduce inconsistencies in the current context. Thus when a change is made EMASON switches to a forward chaining mode. EMASON first begins searching through the current context and compares it with the knowledge base. Any unsubstantiated facts are removed. Unsubstantiated facts are those facts in the context for which the antecedent facts have been withdrawn (likely because of exploration or change

of mind by the user). This procedure is repeated until the current context is consistent with the knowledge base. EMASON then forward chains making new deductions from the changed parameters until no more deductions can be made. The system then switches back to a backward chaining mode and proceeds from the point where the modification was made.

Another important requirement for engineering KBES is the ability to temporarily escape from the system and transfer control to an exterior program, usually some type of applications program. For example, once the KBES has reached a certain point in the solution of a given problem it may be necessary to perform a more refined analysis before continuing. The system should invoke a pre-processor routine to prepare the appropriate data (for a Finite Element program for example). Control would then be transferred to the package until it has completed its task. When that has finished the KBES would regain control and pipe the exterior program's output to a post-processor that would retrieve the relevant information. The KBES would then continue with its problem solving task.

The latter issue raised the question as to whether or not programs which reside outside the KBES should be able to modify the context - the author feels strongly that they should not. It is almost essential that all modifications to the context be done within the KBES. In this manner a record can be kept of all changes that have occurred in the context. Allowing *ad hoc* modifications would make it virtually impossible to maintain correctness. In addition, because of the asynchronous nature of the execution of the external programs, it would make it much harder to ensure that a user's query would be answered from a context that had been fully updated.

4.3.4 Form of the Production Rules

Rules in EMASON were given a simple representation — as lists in LISP. A typical production rule is shown below.

```
(RULE Identifier (IF clauses)
                      (THEN clauses)
                      (Message)
                      (Function)
                      (Attenuation))
```

A production rule is a list that contains seven elements. The first element, the word RULE, is purely cosmetic and serves to improve readability. The second element is the rule identifier. The rule identifier is used to refer to the rule at high levels of abstraction where it is not necessary to carry the complete list. The third element contains the If-clauses. This is the antecedent side of the production rule and contains a list of conditions. The consequent side, contain the actions, appears in the fourth element. Element number five contains a message, if any, to be printed if the rule containing it is fired. The last two elements in the list are used to estimate the measure of belief of the conclusions of a specific production. Measures of belief are discussed in Section 4.3.8.

The antecedent and consequent sides of a production are lists of clauses. The clauses have the following form.

(Building is a post disaster building)

(~Units are light weight)

(@ NOT (= (GET_VAL \$MORTAR) 'N))

The '~' in the second clause represents the negation of the condition. It is preferable to attempt to prove facts from a positive point of view rather than a negative. For example;

(IS THIS TRUE —> EAST WALL IS BUILT OF CLAY BRICKS ?)

is preferable to

(IS THIS TRUE —> EAST WALL IS NOT BUILT OF CLAY BRICKS ?)

Thus when the system encounters the '~' it attempts to prove the fact and takes the logical negation of the answer. The '@' in the third clauses syntactically differentiates evaluable expressions from factual or textual expressions and is always followed by a valid piece of LISP code. The meaning of third clauses is that it is true if the mortar type is not "N". The binding of variables in EMASON is handled through the use of evaluable expressions. An example of a variable binding is given in the following clause, where the term BINDER indicates that the variable following it is to be bound to the the expression following the variable.

(@ BINDER (QUOTE E_CON) (* 5000.0 (SQRT FC_PRIME)))

All rules have, associated with them, permanent statistics. The book keeping procedures are intended for use with the knowledge acquisition module as the knowledge base grows and develops. Some typical statistics are shown below.

(Rule-Identifier Stats

Date last modified < >
 Author < >
 Times considered < >
 Times used < >
 Percentage use < >)

4.3.5 The Explanation Module

One of the most highly touted features of KBES has been their ability to reason about themselves. Reasoning about self is needed so that the user can interact with the KBES and question its results, operations, deductions, and current beliefs. Currently most KBES only touch on the explanation aspect of reasoning. This is due to the fact that most knowledge bases are shallow and broad as opposed to deep and narrow. A deep knowledge base would give a KBES the ability to look at its line of reasoning and understand it. Research being done on Reason Maintenance Systems [20] should address the issues of proper explanations. EMASON concentrates on the explanation aspect of reasoning about self. Explanation is the by-product of the ability to recreate the inference paths.

All communications with EMASON are handled through the explanation module. A summary of the features implemented in the explanation module is given below. Examples of their use is shown in Chapter 6.

- 1) CHANGE - Change the value of variable. This procedure maintains consistency in the context.
- 2) CURRENT - Display the current rule or hypothesis being considered.
- 3) GET@ - Display from the current context the known facts, the rules used or the bindings of specific variables.
- 4) HAVE YOU USED RULE - Query the system as to whether a specific production has been used.
- 5) HELP@ - An on line help facility
- 6) HOW - Retrace the inference path to display the evidence that EMASON used to arrive at a specific conclusion or display the evidence that would be necessary to deduce a conclusion.

- 7) RULES LIST - Display the list of rule identifiers from the knowledge base.
- 8) SHOW RULE - Display a specific rule in a pseudo english format.
- 9) STATS RULE - Display statistics on a specific rule.
- 10) STATUS - Display the status of a specific fact, rule or variable.
- 11) WHY - Query the system as to why it needed a specific piece of information or why it currently needs a specific piece of information.

The explanation module lacks some functions that are potentially important. A user might want to query the system as to why it did not require a specific information or why it did not deduce a certain conclusion.

4.3.6 Knowledge Acquisition Tools

Knowledge acquisition tools were implemented as part of EMASON in order to aid in the development and growth of the knowledge base. Direct access to the rule base is permitted only through the knowledge acquisition tools. A password system was implemented in order to demonstrate that access should be limited and controlled.

The function of the knowledge acquisition module is to monitor the rule base, keep statistics on the rules themselves, and serve as a means of modifying the rule base while maintaining its consistency. Ideally, a knowledge acquisition module should be an expert system whose domain is the KBES that it serves. The knowledge acquisition tools implemented in EMASON were very limited and intended solely to facilitate the development of the knowledge base. The tools that were implemented are listed below.

- 1) ADD NEW RULE
- 2) DELETE A RULE
- 3) COMPILE rule base
- 4) WRITE TO TEXT FILE rule base

COMPILE compiles the rule base in order to integrate any modifications made by the knowledge engineer into the knowledge base. WRITE TO TEXT FILE writes the rule base to a text file in a pseudo english format. An editing facility, such as EDIT A RULE was not provided for in EMASON, though it would be possible to construct a powerful rule editor using the LISP structure editor provided in most versions of LISP.

4.3.7 Natural Language Interfaces

All communications with KBES should be passed through a Natural Language Interface (NLI) facility. An NLI allows the user to communicate with the system using conversational format using everyday English. There is a considerable effort underway to develop NLI technology. It is generally regarded that a proper NLI is not possible given the current state of technology and understanding. In the engineering field, as with most of the sciences, communication is generally accomplished using a highly constrained subset of the English language. State of the art AI techniques do permit the development of a NLI based on a highly constrained version of English. The development of an NLI for EMASON was not considered to be within the scope of the research. It was felt that eventually it will be possible to purchase a commercially available NLI that can be modified to suit the user's needs.

4.3.8 Certainty Factors

Most real world tasks defy absolute or categorical analysis due to complexity and lack of complete knowledge. Many KBES provide the user with a numerical measure of the system's belief in a specific hypothesis. This measure of belief is often called a certainty factor.

An obvious way of handling the issue of uncertainty would be to measure the reliability of knowledge as an estimate of probability. Bayes theorem would be used to compute the conditional probabilities of conclusions based on observations.

Several drawbacks have been mentioned to this approach [52]. Bayes Theorem requires a complete set of hypotheses. The nature of knowledge is that new hypotheses are constantly being advanced. This would require the introduction of a *none of the above* hypothesis. In order to given an accurate estimate, Bayes Theorem requires that all the possible outcomes be disjoint, whereas many real world problems can have several different solutions.

$$P(h_i/E) = \frac{P(E/h_i)P(h_i)}{\sum_{n=1}^k P(E/h_n)P(h_n)}$$

where: $P(h_i/E)$ is the probability of h_i
 given that event E has occurred.
 $P(E/h_i)$ is the probability that event
 E has occurred given the
 hypothesis h_i .
 $P(h_i)$ is the probability of h_i
 being true.
 k is the number of probable hypothesis.

Bayes Theorem [2].

MYCIN uses a modified Bayesian approach [52]. All assertions being considered have associated with them a Measure of Belief (MB) and a Measure of Disbelief (MD).

$$MB(h, e) = \begin{cases} 1, & \text{if } P(h) = 1; \\ \frac{\max[P(h/e), P(h)] - P(h)}{\max[1, 0] - P(h)}, & \text{otherwise.} \end{cases}$$

$$MD(h, e) = \begin{cases} 1, & \text{if } P(h) = 0; \\ \frac{\min[P(h/e), P(h)] - P(h)}{\min[1, 0] - P(h)}, & \text{otherwise.} \end{cases}$$

where: $MB(h, e)$ is the measure of belief in h
 given e .

$MD(h, e)$ is the measure of disbelief in h
 given e .

$P(h)$ is the probability of h occurring.

$P(h/e)$ is the probability of h occurring
 given e .

An estimate of the certainty factor, CF , is given as;

$$CF(h, e) = MB(h, e) - MD(h, e)$$

The *a priori* certainty factors $P(h/e)$ and $P(h)$ must be entered at the time the rule is entered into the knowledge base.

Another approach to measuring uncertainty is to use the Theory of Fuzzy Sets [46]. This approach was used by Ishizuka and Yao [32] in the SPERIL-I system. They have extended Dempster's and Shafer's Theory [3] to include the use of fuzzy sets.

Provisions have been made, based on some simple methods outlined by Winston [74], in EMASON to incorporate some method of measuring uncertainty. Currently the system recognizes three simple functions that calculate a measure of belief. MINNER assigns to a conclusion the minimum measure of belief from the list of supporting evidence. MAXER assigns to a conclusion the maximum measure of belief from the list of supporting evidence. BOOLEAN assigns the corresponding values associated with TRUE or FALSE, this was usually 1 or 0, depending on the supporting evidence. Each rule in turn contains an attenuation factor. This number is a measure of the inherent reliability of the process. Once a certainty factor has been calculated it is multiplied by the attenuation factor and the result is assigned to the conclusion(s). A threshold limit, below which a hypothesis was deemed unsupportable, was also included in EMASON. This limit was usually set, somewhat arbitrarily, at 0.2. The area of measuring the belief or disbelief of a specific hypothesis in EMASON needs considerable development.

4.3.9 Implementation Issues

The implementation of MASON resulted in a knowledge base with a growing number of rules and variables. In particular it was found that there was a large increase in the number of variables as the development proceeded into domains requiring some moderately complex calculations. It became quickly obvious that execution time in the inference cycle was going to be a major problem. In order to reduce searching time EMASON requires that the rule base be pre-compiled into an inference network. Each fact and variable in the knowledge base has associated with it a list containing the identifiers of the rules that can derive it. The inference mechanism simply accesses these lists and examines the rules to which they point. Simple modifications to the compiling procedure and inference engine can allow for the prioritization of the knowledge base if it is so desired.

As the MASON rule base grew it became apparent that the compiling procedure was slowing down the development of the rule base. To speed up the knowledge acquisition phase an incremental compiler was implemented. The incremental compiler updates the knowledge base on a rule by rule basis. This was found to be an order of magnitude more

efficient. Presently the maintenance of the MASON knowledge base is similar to that of file management in a computer system. Incremental compiles are done every time a new rule is entered while full compiles are only done periodically. The full compile stage will be eliminated completely when the incremental compiler has been fully tested.

4.4 Summary

The EMASON ES Framework developed is a rule based inference mechanism that supports and maintains knowledge bases having a variety of domains. The system operates predominately in a backward chaining mode, although it can be operated in a forward chaining mode. EMASON handles the many numeric variables that are required for engineering calculations and attempts to maintain the consistency and integrity of the context if the user decides to modify any initial parameters. There is much room for improvement in the area of efficiency in the system, however constraints have dictated that the development of an operating version of MASON and EMASON was paramount. The intent of the EMASON system was to stimulate further research into the application of AI techniques in the domain of Civil Engineering.

Chapter 5

The MASON Knowledge Base

As men journeyed in the east they came upon a plain in the land of Shinar and settled there. They said to one another, 'Come let us make bricks and bake them hard'; they used bricks for stones and bitumen for mortar. 'Come,' they said, 'Let us build ourselves a city and a tower with its top in the heavens, and make a name for ourselves; or we shall be dispersed all over the earth.'

—Genesis 11.2-9.

This chapter will cover the implementation and organization of the MASON knowledge base. MASON is a system used to diagnose the possible failure mechanisms in masonry walls and was implemented using the EMASON ES framework. The MASON system consists of a collection of production rules and special functions that are specific to the masonry domain. The organizational structure of the rule base evolved to its present state during the implementation process, therefore the topics are closely related. It should be emphasized that the description of the MASON rule base is, in reality, a 'snap shot' of the knowledge base at a specific point in time. The rule base is in a constant state of flux as more is learned about the MASON knowledge base and the EMASON ES framework.

5.1 Implementation of the Knowledge Base

As it stands the current rule base can be viewed as a 'first draft' of a body of knowledge that is to be expanded on and refined. Several failure mechanisms were chosen from various literature sources [21], [9], [59], and served as a starting point for the implementation of a preliminary rule base. The failure hypotheses chosen were fairly narrow in their scope and were by no means complete. Wherever possible, avenues were created or left open for future generalizations, thus allowing for the broadening of the scope of the rule base. The MASON rule contains very specific failure modes pertaining to specific types of building structures. This should be expanded in the future. The nature of the failure modes will be discussed later in this chapter.

The MASON knowledge base contains six distinct failure modes. They are:

- 1) Failure Type 1: Local bearing failure.
- 2) Failure Type 2: Compression failure of wall.
- 3) Failure Type 3: Wall beam interaction problem.
- 4) Failure Type 4: Wall frame interaction due to lateral loading.
- 5) Failure Type 5: Tension failure due to wall shrinkage.
- 6) Failure Type 6: Cladding distress (not implemented).

Failure types 3, 4, and 5 will be discussed in more detail in Section 5.3.

Proceeding backwards from a specific failure hypothesis, rules were encoded to characterize the specific failure mode. Working from a highly constrained domain, the knowledge required began to broaden out to encompass the more general domain of structural engineering. A particular example of this was the implementation of Failure Type 4. The number of rules needed to encode this failure mode grew quickly and resulted in the generation of 40 new rules, and is still by no means complete within its scope. The specific failure mechanism was taken from pages 6-26 to 6-31 of Drysdale and Suter [21]. As the knowledge was encoded it was noticed that a large body of information was required on the various properties of masonry and masonry assemblages and obtained from various sources [43], [13]. Finally the scope had broadened such that knowledge was being obtained from such sources as the National Building Code and the Canadian Standards Association [43], [11], [14]. The phenomenon of a rapid broadening of the scope was common to all the failure modes, however it was observed that the more general rules required by the specific failure rules began to overlap. What was being created was a larger, slightly more general, body of knowledge whose domain was the field of building construction and was not restricted to that of masonry.

This large volume of information and number of rules forced an *ad hoc* organizational structure upon the knowledge base. Initially all the rules were given simple numeric identifiers that increased sequentially according to the order in which they were created. As the rule base became larger a more formal organizational structure was found necessary. The structure will be discussed in Section 5.2. It should be emphasized that the organization of the knowledge base was implemented to speed up the knowledge acquisition phase and to provide a framework for the end user.

5.2 Organization of the Knowledge Base

This section discusses the organization of the MASON knowledge base. The MASON knowledge base was created in order to be compatible with the EMASON ES Framework. It consists of a collection of production rules. Production rules were discussed in Section 3.3.2.

The organization of the MASON knowledge base is characterized by five distinct classes of rules. The five classes are;

- 1) Special Rules.
- 2) General Rules
- 3) Property Rules.
- 4) Masonry Rules.
- 5) Failure Rules.

Special Rules are rules that are 'fired' at the outset of every MASON session, and are used to initialize the context. General Rules serve to focus the attention of the system on specific failure mechanisms. Property Rules and Masonry Rules contain general knowledge and are intended to be utilized by the General and Failure class rules. Failure Rules represent a specific failure hypothesis. The schematic organization of the MASON knowledge base is shown in Figure 5.1. The five classes of rules are discussed in more detail in the following sections.

5.3 Special Rules

Special Rules are rules that are fired automatically at the beginning of every session with MASON. Special rules are fired regardless of the contents of the current context. These rules are used to preset variables, such as geographically dependent information. They are also used to automatically load domain specific files and information. The current implementation of EMASON differentiates Special Rules syntactically from other rules in the knowledge base. The differentiation is achieved by using a unique rule identifier. EMASON contains a procedure called EXECUTE_SPECIAL_RULES. This procedure fires all rules containing a user defined sequence in their identifier. Currently all Special Rules

Figure 5-1: Organization of the MASON Knowledge Base.

in MASON contain the sequence of characters 'SR_' in their identifiers. Examples of a few Special Rules are shown below ¹

```

Rule:  SR_1
      IF
        Value of parameter location is unknown
      THEN
        Prompt the user for the value of location
        and load climatic data
  
```

¹ Note that the rules presented in this chapter are in a stylised format and have been manually translated. Actual rules in MASON are specified using a LISP-like syntax.


```

Rule:   SR_2
        IF
          True
        THEN
          Load appropriate files containing masonry parameters

```

Rule SR_1 states that if the location of the structure is not currently known then the user should specify the location. Once the location has been specified then the appropriate climate data is retrieved from the National Building Code [44]. Rule SR_2 is fired every time it is considered. This rule is used to load files that contain items such as standard block sizes, nominal dimensions of bricks, mortar types and strengths, and other information relevant to masonry construction. A list of the Special Rules in the MASON system is given in Appendix D, Section 1.

5.4 General Rules

General Rules, unlike the Special class of rules, receive no special attention from the MASON system. General Rules were intended to focus the systems attention to a more specific type of failure mechanism. These rules, containing a very broad knowledge about failures in masonry walls, are usually invoked near the beginning of a session. The General Rules were intended to be fired with a relatively empty context. These rules could be viewed as Meta-rules in that specific failure mechanisms are either investigated or ignored depending on their firing. The General class of rules are predominately heuristic in nature, and were intended to incorporate the knowledge of an expert engineer. The intent is to capture the expert's ability to quickly eliminate failure types and to focus on a few relevant modes of failure.

General Rules in MASON are characterized by the characters 'GR' in the rule identifier. Unlike the special rules the characteristic features of the rule identifiers are not used by the system. A few General Rules are shown below.

```

Rule:   GR1
        IF
          Wall is cracked and
          Wall is an infill and
          Wall is non load bearing
        THEN
          Investigate frame wall interaction and
          Investigate temperature and shrinkage effects

```

Rule GR1 provides for a narrowing of the domains scope from all possible failure modes to two relatively broad areas of investigation. The rule requires an extremely cursory inspection of the damaged wall and very little specific knowledge of the building in question.

Rule: GR11

IF

Investigating frame wall interaction and
Cracks appear near the center of the infill panel and
Cracks pass through the mortar

THEN

Investigate the possibility that cracking
has been caused by lateral loading of the frame

Based on a more detailed examination of the wall Rule GR11 narrows the focus somewhat. The hypothesis is that failure was caused by lateral loading and the interaction of the supporting frame with the infill wall. Rules GR12 and GR13 require a more detailed inspection of the wall in question, however once fired they are able to focus the system's attention to a specific mode of failure, namely Failure Type 4. It should be noted that the two rules are almost identical. Rule GR13, however, provides more specific evidence for Failure Type 4. This difference is reflected by the fact that Rule GR13 has a higher attenuation factor than Rule GR12.

Rule: GR12

IF

Investigating the possibility that cracking
has been caused by lateral loading of the frame and
Crack orientations are diagonal to the panel
There is no evidence of crushing at the
diagonally opposite corners

THEN

Investigate failure type 4

Rule: GR13

IF

Investigating the possibility that cracking
has been caused by lateral loading of the frame and
Crack orientations are diagonal to the panel
There is evidence of crushing at the
diagonally opposite corners

THEN

Investigate failure type 4

The four rules above demonstrate how the General class of rules narrow the scope of the problem to a relatively few hypotheses. A complete list of the General Rules in MASON is given in Appendix D, Section 2.

5.5 Property Rules

Like the General Rules, Property Rules receive no special treatment from the system. Property Rules are domain specific rules that provide knowledge about properties of engineering materials. These rules were designed to handle the mechanics as well as properties of such materials as steel, concrete, and masonry. In the implementation of the MASON knowledge base it was noticed that these rules seemed to belong to the more general domain of building construction, rather than the subset domain of masonry walls. This distinction is particularly important for organizational and pedagogical reasons. The nature of the knowledge contained within these rules was a combination of general knowledge, empirical relations, mathematical expressions, and heuristic rules of thumb. The knowledge for these rules was obtained from such sources as building codes and standards. The following set of rules was derived from ACI Committee 209 [1]

Rule: PR2
 IF
 Supporting structure is concrete, and
 Building is more than one year old
 THEN
 Assume that 90% of the ultimate shrinkage has taken place
 $\epsilon_c = \epsilon_T = 0.9\epsilon_{ult}$

Rule: PR6
 IF
 Supporting structure is concrete, and
 Building is not more than one year old and
 Concrete has been moist cured
 THEN
 Calculate shrinkage according to ACI Committee 209
 $\epsilon_c = \epsilon_T = \epsilon_{ult} T_{days} / (T_{days} + 35)$

Property Rules also contained some heuristic knowledge. An example is rule PR1 (if the value of f'_c is not available then the concrete strength is assumed to be 25 Mpa)

Rule: PR1
 IF
 Supporting structure is concrete and
 f'_c is not known
 THEN
 Prompt user for the value of f'_c
 using a default of
 $f'_c = 25 \text{ Mpa}$

Rule PR18 is intended to retrieve the value of f'_m from Table 3 in clause 4.3.3.7 of CSA-S304.M78 [13]

```

Rule:   PR18
      IF
        Units are block, and
        Mortar is type N
      THEN
        Get  $f'_m$  for block, having
         $f'_c$  and mortar type N
  
```

Rule PR22 retrieves the value of f'_c for a given type of unit. The user is first prompted for an estimate of f'_c . If the user is unable to provide MASON with the required information then it assumes a value

```

Rule:   PR22
      IF
        Value of  $f'_c$  is not known
      THEN
        Prompt user for the value of  $f'_c$ ,
        using a default of
        20 Mpa if units are block
        or 40 Mpa if units are brick
  
```

Rule PR23 retrieves the physical properties of a given block. The list of properties was taken from page 2-7 of Drysdale and Suter [21].

```

Rule:   PR23
      IF
        Units are block, and
        Properties are unknown
      THEN
        Prompt user for the type of block
        using a default of
        Hollow Block and
        Load the appropriate properties
  
```

The functions that retrieve and search through the above mentioned tables, and the tables themselves, are loaded into the system at the beginning of each session, by the firing of Special Rule SR_2. Property Rules in MASON are categorized by the characters 'PR' in the rule identifier. A list of the Property Rules contained in MASON is given in Appendix D, Section 3.

5.6 Masonry Rules

Masonry Rules were originally intended to provide the system with a general knowledge of masonry properties, behavior and construction techniques. These rules are more domain specific than the Property or General Rules, however they contained no information about specific failure mechanisms. The information for these rules was obtained from various sources [13], [21], [50], [59]. Examples of two simple Masonry Rules are given below:

```
Rule: ID002
      IF
        Wall has been grouted
      THEN
        Width of wall equals width of unit
        i.e.  $w_{wall} = w_{unit}$ 
```

```
Rule: ID003
      IF
        Wall has not been grouted.
      THEN
        Width of wall equals twice the face
        shell thickness of unit
        i.e.  $w_{wall} = w_{faceshell}$ 
```

As the implementation of the MASON knowledge base proceeded, the role of the Masonry Rules was expanded to include rules that contained knowledge about the analysis and design of building structures. Strictly speaking these new rules should have been classified under a newly created class of rules but it was more convenient to combine them with the Masonry Rules.² The following rule is an example of these. The source for this rule was Section 4 of the National Building Code [43].

```
Rule: ID009
      IF
        There is evidence of an earthquake
      THEN
        Calculate Design Earthquake Force
        from NBCC 4.1.9.1 (4)
        i.e.  $V = ASKIFW$ 
```

Rule ID009 calculates the minimum lateral seismic force the building in question has been designed to resist. The following rules calculate the values needed to estimate V

² Future implementations of MASON will include another class of rule for this type of knowledge.

Rule: ID010
 IF
 There is evidence of an earthquake
 THEN
 Calculate Seismic Response factor
 from NBCC 4.1.9.1 (5)
 i.e. $S = 0.5/\sqrt{T}$
 but not to exceed 1.00

Rule: ID011
 IF
 There is evidence of an earthquake, and
 Building is moment resisting frame
 THEN
 Calculate Fundamental Period
 from NBCC 4.1.9.1 (6)
 i.e. $T = 0.1N_{o_storeys}$

Rule: ID012
 IF
 There is evidence of an earthquake, and
 Building is not moment resisting frame
 THEN
 Calculate Fundamental Period
 from NBCC 4.1.9.1 (6)
 i.e. $T = 0.09h_n/\sqrt{D}$

The information for the evaluation of the stiffness parameter K was taken from Table 4.1.9.A and clause 4.1.9.1 (7) of the National Building Code [43]. This table was translated into six rules one of which is given below.

Rule: ID019
 IF
 There is evidence of an earthquake, and
 Building has a dual structural system, and
 Building is moment resisting frame, and
 Building has flexural wall, and
 System interaction uses relative rigidities
 THEN
 $K = 0.7$

The importance factor was determined from clause 4.1.9.1 (9) of the National Building Code. This was translated into the following rules:

Rule: ID013
 IF
 There is evidence of an earthquake, and
 Building is post-disaster
 THEN
 Set Importance Factor
 i.e. $I = 1.3$

Rule: ID014

IF

There is evidence of an earthquake, and
Building is not post-disaster

THEN

Set Importance Factor

i.e. $I = 1.0$

The knowledge for the Foundation Factor, F , was obtained from Table 4.1.9 b from clause 4.1.9.1 (10) of the National Building Code [43]

Rule: ID015

IF

There is evidence of an earthquake and
Foundation on rock, or
dense to stiff soil

THEN

set Foundation Factor

i.e. $F = 1.0$

The value for W in Rule ID009, the weight of the building, is obtained by querying the user. If it is so desired, a set of rules can be encoded to estimate the building weight, perhaps by querying the user for the number of floors, building dimensions, and the structural materials used. The Acceleration Ratio, A , is obtained, for the given location of the building, at the outset of the session when Special Rule SR_1 is executed. A list of Masonry Rules is given in Appendix D, Section 4.

5.7 Failure Rules

Failure Rules were highly domain specific and contain knowledge about specific hypotheses. Each sub-class of Failure Rules represents a specific failure mechanism. These rules tended to be highly mathematical in nature, although some of the knowledge contained in them was heuristic or empirical. An example of a heuristic Failure Rule is Rule ID109³. The rule of thumb applied to generate this rule was that column stresses in most normal concrete structures range between 30 and 45 percent of the concrete strength, f'_c . The following sections will examine the MASON failure modes in more detail.

³ The identifier which was numbered in the 100-199 range indicates that this rule pertains to Failure Type 1.

Rule: ID109

IF

Should investigate Failure Type 1. and
Supporting structure is concrete

THEN

Prompt user for the value of the stress in
column, using a default of
45% of the concrete strength

$$\text{i.e. } \sigma_{\text{column}} = 0.45 f'_c$$

5.7.1 Failure Types 1 and 2

Failure Types 1 and 2 dealt with frame shortening and deflection problems. Failure Type 1 represents failure caused by local bearing stresses that exceed the compressive strength of the wall causing local crushing. This situation would occur when the elastic and/or creep deflections of ceiling elements, such as beams, caused them to rest on the top of the wall. Failure Type 2 is caused by overall frame shortening resulting in the infill panel taking on extra gravity loads. Failure would be caused by exceeding the overall compressive capacity of the wall. These two failure types were implemented at the outset of the development of MASON and were intended to test the system's operation. The resulting models are therefore highly simplistic and unrealistic. The models may be elaborated in the future. The failure conditions are shown in Figure 5.2. The rules representing these two failure types are given in Appendix D, Sections 5 and 6 respectively.

5.7.2 Failure Type 3

Many infill walls are partition walls sitting on top of beams in the supporting structure and most of these infills are non load bearing. There is, however, some structural interaction between the wall and the beam. The tendency is for the supporting beam to deflect away from the infill thus producing arching action in the wall and stress concentrations near the beam supports. This action is shown in Figure 5.3.

Research by Davies and Ahmed [17] has shown the stress distributions vary either triangularly, as a quadratic parabola, or as a cubic parabola depending on the stiffness of the wall relative to that of the beam. These distributions are shown in Figure 5.4.

R is a measure of the relative stiffness of the wall and beam and is calculated from the expression given below

Figure 5-2: Failure Types 1 and 2.

$$R = \sqrt{\frac{H_{wall}^3 t E_{wall}}{I_{beam} E_{beam}}}$$

The maximum stress that occurs near the beam supports is given by:

$$f_m = \frac{C_1 W}{L_{wall} T_{wall}}$$

where: $C_1 = 1 + \beta R$,

$W = w L_{wall}$,

β is a parameter depending
on the wall aspect ratio

Figure 5-3: Failure Type 3.

The following is a selection of rules pertaining to this failure mechanism.

Rule: TD302

IF

Should investigate failure type 3. and

$\sigma_{wall} > \sigma_{allow}$

THEN

Wall stresses at the corner have
exceeded the allowable stress

In order to fire this rule two antecedent conditions must be satisfied. The condition that Failure Type 3 should be investigated can be concluded from GR9. The second condition is that the actual stress at the corners of the wall must exceed the allowable stress. The following rules demonstrate how this condition is met. The variables σ_{wall} and σ_{allow} are obtained from firing rules such as Rule ID309

Rule: ID309

IF

Should investigate failure type 3. and

Beam is flexible

THEN

$\sigma_{wall} = \sigma_{max}/3$

Rule ID306 determines whether or not the beam is flexible.

Figure 5-4: Stress Distributions [21].

Rule: ID306

IF

Should investigate failure type 3, and
 $R \leq 7$, and
 $R > 5$

THEN

Beam is flexible.

The value of R was calculated from Rule ID311.

Rule: ID311

IF

Should investigate failure type 3.

THEN

$$R = \sqrt{\frac{H^3_{wall} t E_{wall}}{I_{beam} E_{beam}}}$$

Parameters such as I_{beam} were calculated using rules of type described by Rule ID117.

Rule: ID117

IF

Supporting structure is concrete, and
 Beam has rectangular cross section.

THEN

Calculate the moment of inertia.
 i.e. $I_{beam} = \frac{H_{beam} W_{beam}^3}{12}$

The maximum stress, σ_{max} is calculated in Rule ID303.

Rule: ID303

IF

Should investigate failure type 3.

THEN

Calculate maximum stress.
 i.e. $\sigma_{max} = C_1 w/t$

The constant, C_1 is evaluated by Rule ID304.

Rule: ID304

IF

Should investigate failure type 3.

THEN

Calculate the value of C_1 .
 i.e. $C_1 = 1 + \beta R$

The parameter β was calculated from an approximation of a graph presented on page 6-24 of Drysdale and Suter [21]. The parameter is dependent on the aspect ratio of the wall being investigated.

Rule: ID313

IF

Should investigate failure type 3 and

$H_{wall}/L_{wall} \geq 0.5$ and

$H_{wall}/L_{wall} < 1.0$

THEN

$$\beta = 2.0 - 0.8H_{wall}/L_{wall}$$

Finally the allowable stress, σ_{allow} , is determined from Rule ID316.

Rule: ID316

IF

Should investigate failure type 3

THEN

Set allowable stress equal to the masonry strength

i.e. $\sigma_{max} = f'_m$

The masonry strength, f'_m , is retrieved by Rules PR15 to PR19. The reader is referred to Rule PR18 in Section 5.5. The above rules show how the failure rules are organized such that a conclusion can be reached. In the next section the implementation of a failure mechanism will be traced from its initial development. The complete list of rules for Failure Type 3 is given in Appendix D, Section 7.

5.7.3 Failure Type 4

Consider a frame with an infill wall subjected to lateral loading. The infill wall will contribute to the stiffness of the supporting frame. Failure is likely to occur in the infill panel under extreme loading conditions, such as an earthquake. Depending on the aspect ratio of the panel, failure is likely to be due to tensile splitting of the masonry or shearing of the horizontal mortar joints [15]. Failure Type 4 suggests that lateral loading has caused the failure of the wall. The infill has acted as a compression strut and the shear capacity of the wall has been exceeded causing cracking to appear near the center of the panel. If the infill was improperly detailed crushing failure could also occur at the corners of the panel due to high local stresses or hammering action of the the frame on the wall. This type of failure is shown in Figure 5.5. The scope of this failure type was limited to single storey frames with infill walls.

The general heuristic rules that serve to focus the systems attention on Failure Type 4 were outlined in Section 5.4. Briefly stated they query the user as to whether or not diagonal cracks appear near the center of the infill panel. If so then a more detailed analysis

Figure 5-5: Failure Type 4.

is undertaken. If the lateral load exceeds the capacity of the infill wall then it was assumed that failure was due to lateral loading. This condition was summarised by Rule ID402.

Rule: ID402

IF

Should investigate failure type 4 and

$V_{wall} > V_{allow}$

THEN

Lateral Load exceeds wall's capacity

It was then necessary to estimate the amount of load that the wall carries. The amount of load carried by the wall will vary with different types of structures. The following rules were based on equations taken from Drysdale and Suter [21], pages 6-29 and 6-30.

Rule: ID403

IF

Should investigate failure type 4, and

Structure is a moment resisting frame, and

Structure is pinned at the base, and

$No_storeys = 1.0$.

THEN

$$V_{frame} = \frac{12EI}{H^3} \frac{k}{2k+1} \Delta$$

Rule: ID404

IF

Should investigate failure type 4 and
Structure is a moment resisting frame and
Structure is fixed at the base and
 $No_storeys = 10$

THEN

$$V_{frame} = \frac{12EI}{H^3} \frac{6k+1}{3k+2} \Delta$$

Rule: ID414

IF

Should investigate failure type 4 and
Structure is a pinned frame and
Structure is tension braced and
 $No_storeys = 10$

THEN

$$V_{frame} = k \Delta$$

The variable k was also dependent on the type of structure. For a moment resisting frame k was calculated using Rules ID415 and ID416

Rule: ID415

IF

Should investigate failure type 4 and
Structure is moment resisting frame and
 $No_storeys = 10$

THEN

$$k = \frac{I_{beam} H_{frame}}{I_{column} L_{frame}}$$

Rule: ID416

IF

Should investigate failure type 4 and
Structure is a pinned frame and
Structure is tension braced and
 $No_storeys = 10$

THEN

$$k = \frac{A_{brace} E L_{frame}^2}{(L_{frame}^2 + H_{frame}^2)^{1.5}}$$

The lateral deflection of the frame Δ , was encoded into Rule ID417.

Rule: ID417

IF

Should investigate failure type 4 and
 $No_storeys = 10$

THEN

$$\Delta = \frac{1.2 V_{allow} H_{frame}}{L_{frame} W_{wall} G_{wall}}$$

The lateral load capacity of the wall, V_{allow} , was calculated using Rule ID418.

Rule: ID418

IF

Should investigate failure type 4 and
 $No_storeys = 10$

THEN

$$V_{allow} = L_{wall} W_{wall} V_{allow}$$

The allowable shear stress in the masonry assemblage was calculated using Masonry Rules ID007 and ID008. The lateral load assumed to be taken by the wall was the difference in the actual load and the contribution of the frame. If this value was greater than V_{allow} , then it was assumed that Failure Type 4 was the appropriate mode of failure.

Rule: ID411

IF

Should investigate failure type 4 and
 Structure is a moment resisting frame and
 Cracks are not confined to the end panels

THEN

$$V_{wall} = \frac{V - V_{frame}}{No_bays}$$

Rule: ID407

IF

Should investigate failure type 4 and
 Structure is a pinned frame and
 Structure is not tension braced, and
 Cracks are not confined to the end panels

THEN

$$V_{wall} = V / No_bays$$

In implementing this failure mode only earthquake forces were considered. Wind loading can easily be included by the addition of a few more rules. If the structure in question was not instrumented it would be difficult for the user and the system to estimate the magnitude of the lateral load that was acting on the structure. The design earthquake forces were chosen as an estimate of the lateral load in lieu of more accurate estimates. The lateral load V , was calculated from Rule ID009 which is described in more detail in Section 5.6. The above selection of rules demonstrates how codes and specifications as well as some expert knowledge can be codified into production rules. A complete list of rules describing failure type is given in Appendix D, Section 8.

5.7.4 Failure Type 5

Unit shrinkage and thermal movements are two of the major problems that face designers of walls containing masonry units. Failure Type 5 addresses the problem of shrinkage

of masonry units in walls. Although the failure mode represented by this type of failure is narrow (it deals only with the shrinkage of long walls) it can be expanded. In the future, the scope of this Failure Type could be expanded to include more complicated problems that require a more elaborate analysis.

Consider a long, straight masonry wall with restraints at both ends. Shrinkage of the units would cause the wall to shrink, however the end restraints would resist any movement thus producing tension in the wall. This is shown in Figure 5-6.

Figure 5-6: Failure Type 5.

If the tensile capacity of the wall was exceeded then Rule ID502 would conclude that failure was due to shrinkage.

Rule: ID502

IF

Should investigate failure type 5, and

$\sigma_{wall} > \sigma_{allow}$

THEN

Shrinkage has caused tensile stresses in the wall that have exceeded the tensile capacity of the wall.

The stress in the wall, σ_{wall} , was dependent on the wall shrinkage and was calculated

as:

$$\sigma_{wall} = \frac{\Delta_{wall} E_{wall}}{L_{wall} - \frac{2 Area_{wall} E_{wall}}{E_{connection}}}$$

where $\Delta_{wall} = L_{wall} \epsilon_{sh}$

The shrinkage rates for the units were taken from Table 2 on page 3-21 of Drysdale and Suter [21]. Only concrete block was considered for this failure type

Rule ID508

IF

Should investigate failure type 5 and
Block has been High Pressure steam cured and
Block was saturated when laid

THEN

Get unit shrinkage for block HP Saturated
with specific aggregate type

Rule: ID511

IF

Should investigate failure type 5 and
Block has not been High Pressure steam cured
i.e. Block has been High Temperature
steam cured, and
Block was not saturated when laid
i.e. Block was laid at 70%
Relative Humidity

THEN

Get unit shrinkage for block HP Saturated
with specific aggregate type

The area of the wall was calculated as the height of the wall multiplied by the face shell thickness of the unit. The height depended on whether or not the wall contained openings and whether the cracks appeared near them.

Rule: ID518

IF

Should investigate failure type 5, and
Wall contains openings, and
Cracks appear near the openings

THEN

$$H_{wall} = Height_{wall} - Opening_{nrc}$$

Rule: ID519

IF

Should investigate failure type 5, and
Wall does not contain openings, and

THEN

$$H_{wall} = Height_{wall}$$

The allowable tensile stress for a specific wall was taken from Table 7 on page 36 of CSA S304-M78 [13]

Rule: ID512

IF

Should investigate failure type 5 and
Units are block and
Bond pattern is stack bond and
Mortar is type N

THEN

$\sigma_{wall} = 0.22 \text{ Mpa}$

A complete list of rules that characterize Failure Type 5 is given in Appendix D, Section 9. This failure mechanism represents a starting point for a potentially much broader failure mode, namely that of thermal and moisture deformations. M. A. Mohamed [41] provided an excellent overview of low rise masonry walls subjected to thermal and moisture movements. The first part of that thesis summarized existing literature that dealt with moisture and temperature effects, while the second part analyzed walls subjected to these effects using finite element analysis. Although this document was not extensively used in the development of this failure mode it should serve as a basis for any future extensions of Failure Type 5.

5.7.5 Failure Type 6

Failure Type 6 was not implemented. This failure mechanism would deal with cladding distress problems. Cladding distress is a growing concern given the state of building practice and there exists a considerable amount of literature on this subject. This area is perhaps one of the most important directions for the future development of MASON. Of particular interest would be the attempt to interface the MASON system with the VENEER program developed by G. A. Fenton [23].

5.8 Summary

MASON was a system developed using the EMASON ES framework. The task of MASON was the rule based damage assessment of masonry walls. It consisted of a set of production rules that dealt with failures in masonry. These rules were organized into five distinct classes. Each class of rule was characterized by an increased focusing of domain knowledge. Special Rules, for example, contain no knowledge about masonry failures, while

Failure Rules contain only knowledge about specific masonry failures. MASON was implemented with five types of failure conditions. These conditions were implemented for three purposes. The first objective was to test the EMASON system. The second objective was to demonstrate how rule based systems could be applied to engineering problems. The final objective was to implement a system that would serve as a basis for future development in rule based damage assessment of masonry walls. The type of knowledge used in developing the MASON rule base ranged from general knowledge of building structures to specific knowledge about masonry failures. This information was drawn from various sources, such as textbooks, specifications and research papers.

Chapter 6

Sample Interactions with MASON

The question and answer method seems to be suitable for introducing almost any one of the fields of human endeavor that we wish to include. We do not wish to penalize the machine for its inability to shine in beauty competitions, nor to penalize a man for losing a race against an airplane. The conditions of our game make these disabilities irrelevant. The "witnesses" can brag, if they consider it advisable, as much as they please about their charms, strength or heroism, but the interrogator cannot demand practical demonstrations.

—A. M. Turing
Computing Machinery and Intelligence

6.1 Introduction

This chapter presents a few sample interactions with the MASON system. The examples used were somewhat contrived but were intended to demonstrate the capabilities and operation of the MASON system. Three distinct failure types, namely Type 3, Type 4, and Type 5 were chosen to demonstrate the system.

6.2 Sample Session One

Consider a structure with a relatively high block wall around the exterior. The walls are load bearing. In order to accommodate the delivery of goods a five meter opening has been left in the wall to serve as a loading door. A reinforced concrete beam has been placed above this loading door to support the wall. Cracks have appeared at the top corners of the door near the beam supports. The situation is shown in Figure 6.1. The beam was assumed to be pin supported.

Figure 6-1: Case for sample session one.

The following values were used for this example and provided to the MASON system via user response.

Location = Ottawa
 Name of Building = K-Mart
 Block = Hollow

$\gamma'_b = 10.0 \text{ Mpa}$
 Mortar = Type N
 Grouted = No
 $f'_c = \text{Unknown}$
 $H_{\text{wall}} = 4500.0 \text{ mm}$
 $L_{\text{wall}} = 5000.0 \text{ mm}$
 $h_{\text{beam}} = 350.0 \text{ mm}$
 $b_{\text{beam}} = 190.0 \text{ mm}$
 $l_{\text{beam}} = 5000.0 \text{ mm}$
 $W_{\text{roof}} = 3.0 \text{ Kpa}$

The interaction is given below.¹

% Start the mason system, the knowledge base is,
 % loaded automatically.

6 lisp> mason
 loading rule base
 getting there
 getting further
 getting even further
 almost there
 done

Hi...

When you wish to initiate a run type RUN.

% Begin a run of the system to start a session.

input>run

Do you wish to enter any initial facts? Y/N

% The system queries the user as to whether or not

¹ The sample outputs in this chapter were slightly stylized to improve their readability. User inputs to the system are in a slanted font while the system's responses are in a sans serif font. Comments are preceded by %.

% there are any initial facts to be entered

input>n

Do you wish to select a hypothesis? Y/N

% The system queries the user as to whether or not

% a hypothesis is to be selected

input>y

Which hypothesis do you wish to select?

% User queries the system as to available hypothesis

input>?

(TYPE1 TYPE2 TYPE3 TYPE4 TYPE5)

For a description type

ELABORATE hypothesis

or Q to quit

Which hypothesis do you wish to select?

% User wishes to examine hypothesis number 3

input>elaborate type3

Wall beam interaction failure.

Which hypothesis do you wish to select?

% User selects hypothesis number 3.

input>type3

Where is the building in question located?

units >NIL

input>Ottawa

% The system loads geographically dependent information

% using rule SR_1. Values for Ottawa are loaded.


```

(VARIABLE > LOCATION = OTTAWA)
(VARIABLE > JAN_TEMP_2HAF_PER = -25 0)
(VARIABLE > JAN_TEMP_1_PER = -27 0)
(VARIABLE > JULY_TEMP_2HAF_DRY = 30 0)
(VARIABLE > JULY_TEMP_2HAF_WET = 23.0)
(VARIABLE > DEG_DAYS_<_18 = 4673)
(VARIABLE > GRD_SNOW_LD = 2 9)
(VARIABLE > 15_MIN_RAIN = 23)
(VARIABLE > 11_DAY_RAIN = 93)
(VARIABLE > ANN_TOT_PCPN = 846)
(VARIABLE > 11_IN_10_WIND = 0.37)
(VARIABLE > 11_IN_30_WIND = 0.37)
(VARIABLE > 11_IN_100_WIND = 0.37)
(VARIABLE > SEISMIC_ZONE = 2)
(VARIABLE > ACCELER_RATIO = 0.04)

```

```

% System loads masonry parameters and files specific to
% this domain

```

Masonry parameters loaded

CISC handbook loaded

```

(VARIABLE > E_STEEL = 200000)

```

```

% The user is asked to supply an identifier for the
% wall in question.

```

Please enter a value for the variable:

\$WALL

units > NIL

```

% The user queries MASON as to the expected response

```

input>?

?

You are expected to respond in one of the following manners:

Some measure of belief pertaining to query.

A value for a variable.

A valid system command (available through MENU).

A valid LISP expression.
 MENU to retrieve the menu.
 Q to exit the system.
 HELP@ [topic].

input>east_wall

(VARIABLE > \$WALL = EAST_WALL)

To what degree is this true: EAST_WALL IS CRACKED ? $0 \leq CF \leq 1$

% User responds to the above question that the east
 % wall is definitely cracked and enters the maximum
 % amount of certainty.

input>1.0

To what degree do you believe that EAST_WALL IS AN INFILL ? $0 \leq CF \leq 1$

input>1.0

How certain are you that: EAST_WALL IS NON LOAD BEARING ? $0 \leq CF \leq 1$

% User responds that the east wall is load bearing
 % and therefore enters the minimum amount of certainty
 % in response to the systems query.

input>0.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT FRAME WALL INTERACTION
 SHOULD BE INVESTIGATED)

(RULE GR10 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

To what degree is this true: CRACKS APPEAR AT THE CORNERS OF THE WALL ?
 $0 \leq CF \leq 1$

% User wishes to examine rule GR10, the system
 % responds by displaying the desired rule.

input>show rule gr10

rule $\Rightarrow$ GR10

2

1.0

1.1

1.25

1.4

1.6

1.8

2.0

2.2

2.5

2.8

3.2

3.6

4.0

```

IF
($WALL IS CRACKED)
($WALL IS AN INFILL)
(~$WALL IS NON LOAD BEARING)
THEN
(FRAME WALL INTERACTION SHOULD BE INVESTIGATED)
MESSAGE
NIL
FUNCTION
(MINNER)
ATTENUATION_FACTOR
(1.0)

```

T

% The user wishes the current system query repeated.

input>repeat query

To what degree is this true: CRACKS APPEAR AT THE CORNERS OF THE WALL ?

$0 \leq CF \leq 1$

T

input>1.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT WALL BEAM INTERACTION
SHOULD BE INVESTIGATED)

(RULE GR8 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

To what degree do you believe that THE UNITS ARE CRACKED ? $0 \leq CF \leq 1$

input>1.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT SHOULD INVESTIGATE
FAILURE TYPE 3)

(RULE GR9 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

% The system determines that based on the cursory

% evidence available that failure type 3 should be

% examined. Failure type 3 deals with wall beam

% interaction type failures.

Are the units brick or block?

input>block

(VARIABLE > \$UNITS = BLOCK)

Could you supply the type of blocks used? Y/N

Choices are:

(HOLLOW 175_SOLID SOLID_PIER)

input>y

input>hollow

% The system retrieves the appropriate physical

% properties for the block used in the building.

(VARIABLE > FACE_SHELL_THICKNESS = 32.0)

(VARIABLE > WIDTH_UNIT = 190.0)

(VARIABLE > EQ_WIDTH_UNIT = 106.0)

(VARIABLE > WALL_MASS = 223.0)

(VARIABLE > WALL_MASS_LT = 181.0)

(VARIABLE > *PROPS* = T)

Is the mortar type N, M or S?

input>n

% More specific information is needed about the

% masonry before any parameters can be computed.

% MASON queries the user to elicit information

% on the mortar and block strength in order to

% calculate f'_m and E_{wall} .

(VARIABLE > \$MORTAR = N)

Could you provide an estimate of the compressive strength of the units? Y/N

input>y

input>10.0

(VARIABLE > FB_PRIME = 10.0)

(VARIABLE > FM_PRIME = 5.4)

(TAKEN FROM CSA S304-M78 14-3-3-7)

(VARIABLE > E_WALL = 5400.0)

(E_WALL HAS BEEN CHOSEN AS THE LEAST OF $1000 \cdot FM_PRIME$ OR 20000 MPA)

% The system now prompts the user for specific
% information about the geometry of the wall:

What is the height of the wall?

units > MM

input > 4500.0

(VARIABLE > HEIGHT_WALL = 4500.0)

Please enter a value for the variable:

\$BUILDING

units > NIL

input > k_mart

(VARIABLE > \$BUILDING = K_MART)

How certain are you that: SUPPORTING STRUCTURE OF K_MART IS STEEL ?

$0 \leq CF \leq 1$

% User queries the system as to the current rule
% that is being considered. The rule is later
% examined.

input > current rule

ID029

input > show

rule $\Rightarrow$ ID029

IF

(SUPPORTING STRUCTURE OF \$BUILDING IS STEEL)

THEN

(I_BEAM $\leftarrow$ (GET_CISC_HDBK DESIGNATION IX))

MESSAGE

NIL

FUNCTION

(MINNER)

ATTENUATION_FACTOR

(1.0)

T

% Since the building structure is concrete the

% appropriate responses are input.

input>0.0

How strongly do you believe that: SUPPORTING STRUCTURE OF K_MART IS
CONCRETE ?

$0 \leq CF \leq 1$

input>1.0

% The system requires some fairly specific data on

% the properties of the beam supporting the wall

% in order to calculate the relative stiffness of

% the beam and the wall.

To what degree is this true: BEAM HAS A RECTANGULAR CROSS SECTION ?

$0 \leq CF \leq 1$

input>1.0

What is the width of the beam

units >MM

input>??

??

Width of beam or floor

input>current rule

ID117

input>current hypothesis

TYPE3

input>show

rule $\Rightarrow$ ID117

IF

(SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE)

(BEAM HAS A RECTANGULAR CROSS SECTION)

THEN

(I_BEAM $\leftarrow$ (WIDTH_BEAM * HEIGHT_BEAM ** 3 / 12.0))

MESSAGE

(AN ESTIMATE OF THE STIFFNESS OF THE BEAM IS $B \cdot H^{**3} / 12$)

FUNCTION

(MINNER)

ATTENUATION_FACTOR

(1.0)

T

input>repeat query

What is the width of the beam

units >MM

T

input>190.0

(VARIABLE > WIDTH_BEAM = 190.0)

What is the total depth of the beam?

units >MM

input>350.0

(VARIABLE > HEIGHT_BEAM = 350.0)

(VARIABLE > I_BEAM = 6.7885416E8)

(AN ESTIMATE OF THE STIFFNESS OF THE BEAM IS $B \cdot H^{**3} / 12$)

To what degree do you believe that CONCRETE IS NORMAL WEIGHT CONCRETE ?

$0 \leq CF \leq 1$

input>1.0

% The system requires the strength of the concrete in
 % the supporting structure, however the user is unable
 % to provide this information therefore MASON assumes
 % that the concrete strength is 25 Mpa.

Could you supply a value for fc_prime
 The ultimate strength of the concrete Y/N

input>n

All right I will assume one

(VARIABLE > FC_PRIME = 25)

(VARIABLE > E_CON = 25000.0000002)
 (TAKEN FROM CLAUSE 16-4-1 CSA 123-3-M77)

(VARIABLE > E_BEAM = 25000.0000002)

(VARIABLE > R = 7.4456937)

% The measure of the relative stiffness between the wall
 % and the supporting beam has been calculated. MASON finds
 % that the beam is very flexible indicating a triangular stress
 % near the end supports of the beam.

(TRIANGULAR STRESS DISTRIBUTION)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT BEAM IS VERY FLEXIBLE)
 (RULE ID305 USED)
 (CERTAINTY FACTOR $\Rightarrow$ 1.0)

How long is the wall?
 units >MM

% User wishes a description of the variable whose value
 % is being requested.

input>??

??

Length of wall in question

% Here the user asks the system to describe how it
 % arrived at the value for the quantity R.
 % MASON responds by providing the user with supporting
 % evidence that led to the application of the
 % equation that allows R to be bound as well as the
 % equation.

input>how did you get $\Rightarrow$ R

(R CALCULATED FROM:)
 (SHOULD INVESTIGATE FAILURE TYPE 3)
 (@@BINDER R
 (SQRT
 (SQRT
 (/ (* EQ_WIDTH_UNIT (* E_WALL (** HEIGHT_WALL 3))) (* I_BEAM E_BEAM))))))

T

% The consultation is continued.

input>5000.0

(VARIABLE > LENGTH_WALL = 5000.0)
 (VARIABLE > BETA = 1.2799999)
 (VARIABLE > C1 = 10.530488)
 (C1 CALCULATED AS $1 + BETA * R$)

What is the super-imposed load on the beam?
 units >KN/M

input>??

??

Linear load on beam or floor above wall

input>12.0

(VARIABLE > W_LOAD_BEAM = 12.0)

How certain are you that: EAST_WALL HAS BEEN GROUTED ? $0 \leq CF \leq 1$

% The user requests that a list of systems commands be
% displayed.

input>menu

MENU

commands are:

ADD_NEW_RULE

ADD_HYPOTHESIS

CHANGE variable TO new value

COMPILE desired rule base

CURRENT rule or hypothesis

CREATE_RULE_BASE

DELETE_A_RULE

DELETE_A_HYPOTHESIS

DESCRIBE variable

ELABORATE hypothesis

Type C to continue menu or X to exit menu.

input>c

commands are:

GET@ KNOWN FACTS

GET@ RULES USED

GET@ VALUE variable

HAVE [you] USED RULE rule identifier

HELP@ [desired topic]

HOW DID [you get] $\Rightarrow$ some conclusion

HOW COULD [you get] $\Rightarrow$ some conclusion

HYPOTHESIS_LIST

MENU

REPEAT QUERY.

RESET

Type C to continue menu or X to exit menu.

input>x

% The user exits the menu and asks the system to display
% a list of known facts. All the known facts are displayed
% along with the associated measures of belief.

input>get known facts

((BEAM IS VERY FLEXIBLE) 1.0)
 ((CONCRETE IS NORMAL WEIGHT CONCRETE) 1.0)
 ((BEAM HAS A RECTANGULAR CROSS SECTION) 1.0)
 ((SUPPORTING STRUCTURE OF K_MART IS CONCRÉTE) 1.0)
 ((SUPPORTING STRUCTURE OF K_MART IS STEEL) 0.0)
 ((SHOULD INVESTIGATE FAILURE TYPE 3) 1.0)
 ((THE UNITS ARE CRACKED) 1.0)
 ((WALL BEAM INTERACTION SHOULD BE INVESTIGATED) 1.0)
 ((CRACKS APPEAR AT THE CORNERS OF THE WALL) 1.0)
 ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) 1.0)
 ((EAST_WALL IS NON LOAD BEARING) 0.0)
 ((EAST_WALL IS AN INFILL) 1.0)
 ((EAST_WALL IS CRACKED) 1.0)

T

input>repeat query

How strongly do you believe that: EAST_WALL HAS BEEN GROUTED ? $0 \leq CF \leq 1$

T

input>current rule

ID006

input>show

rule $\Rightarrow$ ID006

IF

(\$WALL HAS BEEN GROUTED)

(UNITS ARE LIGHT WEIGHT)

THEN

(WT_WALL $\leftarrow$ (323.0 * 9.8099998 / 1000.0))

MESSAGE

NIL

FUNCTION

(BOOLEAN_FN)

ATTENUATION_FACTOR

(1.0)

T

input>0.0

To what degree is this true: UNITS ARE LIGHT WEIGHT ? $0 \leq CF \leq 1$

input>menu

MENU

commands are:

RULES_LIST

SHOW [RULE rule identifier

STATS RULE rule identifier

STATUS OF rule $\Rightarrow$ rule identifier

STATUS OF fact $\Rightarrow$ a certain fact

STATUS OF variable $\Rightarrow$ variable name

UNITS variable

VARIABLES_LIST

WHY DID [you need] $\Rightarrow$ some fact or variable

WHY DO [you need] $\Rightarrow$ some fact or variable

Q to quit to top level.

? to receives instructions at a given level

Type C to continue menu or X to exit menu.

input>x

% The user queries the system as to why it needs

% to know if the blocks are normal or light weight.

% MASON responds that it needs the information to

% calculate the weight of the wall.

input>why do you need $\Rightarrow$ units are light weight

(UNITS ARE LIGHT WEIGHT NEEDED TO SHOW THAT:)

(@BINDER (QUOTE WT_WALL) (* WALL_MASS_LT (/ 9.8099998 1000.0)))

T

input>0.0

(VARIABLE > WT_WALL = 2.1876299)

(VARIABLE > W_LOAD_WALL = 21.8443347)

(VARIABLE > SIGMA_MAX = 2.1701085)

(SIGMA_MAX CALCULATED AS $C1 * W_LOAD_WALL / WIDTH_WALL$)

(VARIABLE > SIGMA_WALL = 1.0850542)

(VARIABLE > SIGMA_ALLOW = 1.08)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT WALL STRESSES AT CORNERS EXCEED ALLOWABLE STRESS)

(RULE ID302 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

(WALL BEAM INTERACTION HAS CAUSED LARGE LOCAL STRESSES NEAR THE CORNERS OF THE WALL NEAR THE BEAM SUPPORTS THESE STRESSES EXCEED THE WALL COMPRESSIVE STRESSES)

(THERE IS DEFINITE EVIDENCE TO SUGGEST CONCLUDE FAILURE TYPE 3)

(RULE ID301 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

% System concludes that the probable cause of failure

% was due to incompatible stiffnesses. The user begins

% to query the system as to how it achieved its conclusions.

Do you wish another run? Y/N

input>n

input>how did $\Rightarrow$ conclude failure type 3

(CONCLUDE FAILURE TYPE 3 DEMONSTRATED BY:)

(SHOULD INVESTIGATE FAILURE TYPE 3)

(WALL STRESSES AT CORNERS EXCEED ALLOWABLE STRESS)

T

input>how did $\Rightarrow$ should investigate failure type 3

(SHOULD INVESTIGATE FAILURE TYPE 3 DEMONSTRATED BY:)

(WALL BEAM INTERACTION SHOULD BE INVESTIGATED)

(THE UNITS ARE CRACKED)

T

input>how did $\Rightarrow$ the units are cracked

(THE UNITS ARE CRACKED WAS GIVEN)

T

input>why did $\Rightarrow$ concrete is normal weight concrete

(CONCRETE IS NORMAL WEIGHT CONCRETE WAS NEEDED TO SHOW:)

(Q\$^BINDER (QUOTE E_CON) (* 5000.0 (SQRT FC_PRIME)))

T

input>how did you get $\Rightarrow$ e_con

(E_CON CALCULATED FROM:)

(SUPPORTING STRUCTURE OF K_MART IS CONCRETE)

(CONCRETE IS NORMAL WEIGHT CONCRETE)

(Q\$^BINDER E_CON (* 5000.0 (SQRT FC_PRIME)))

T

% The user queries the system as to how the
 % value of e_con could be obtained. The system
 % responds by listing all the necessary facts
 % that must be established and expressions that
 % that need to be evaluated. Facts that are known
 % or expressions that have been evaluated are followed
 % by the word KNOWN.

input>how could you get $\Rightarrow$ e_con

(E_CON CAN BE ESTABLISHED BY SHOWING)

(SUPPORTING STRUCTURE OF K_MART IS CONCRETE) KNOWN

(CONCRETE IS NORMAL WEIGHT CONCRETE) KNOWN

(Q\$^BINDER E_CON (* 5000.0 (SQRT FC_PRIME))) KNOWN

OR

(E_CON CAN BE ESTABLISHED BY SHOWING)

(SUPPORTING STRUCTURE OF K_MART IS CONCRETE) KNOWN

(CONCRETE IS NORMAL WEIGHT CONCRETE) KNOWN

(Q\$^BINDER E_CON

(* 0.0429998

(* (* (GET_VAL W_CON) 1.5) (SQRT (GET_VAL FC_PRIME))))

T

% The word BINDER that appears in the last clause
 % is the function that is used to bind variables
 % to a specific value.

input>q

% The satisfied user now ends the consultation
 % and returns from the Expert System to a
 % LISP shell.

MASON concluded that there was a high probability that cracking was caused by arching action of the wall above the concrete beam. Local stresses developed at the beam supports exceeded the allowable stress in the block wall causing tensile splitting of the units.

6.3 Sample Session Two

Consider a single storey warehouse structure. The structural system consists of a pinned steel frame that has been braced using tension diagonal bracing. There are masonry infill walls along the exterior of the building. The walls are not load bearing. A preliminary field inspection reveals that there are diagonal cracks at the end panels along the short dimension of the building. These cracks passed through the mortar. The corners of these panels also appear crushed. Further investigation reveals that an earthquake has occurred recently. The situation is shown in Figure 6.2.

The following information was provided to the MASON system during the consultation.

Location	=	Ottawa
Name of Building	=	Eaton's Warehouse
Block	=	Hollow
f'_c	=	10.0Mpa
Mortar	=	Type N
Grouted	=	No

Frame = Steel

$H_{bay} = 4500.0mm$

$L_{bay} = 7000.0mm$

Beams = W310 x 86

Columns = W250 x 49

$W_{building} = 3.0Kpa$

Figure 6-2: Case for sample session two.

The sample interaction is shown below.

% User runs system

2 lisp> (mason)
loading rule base
getting there
getting further
getting even further
almost there
done

% The following demonstrates the user selecting an appropriate
% hypothesis to be tested during the consultation.

Hi...

When you wish to initiate a run type RUN.

input>run

Do you wish to enter any initial facts? Y/N

input>n

Do you wish to select a hypothesis? Y/N

input>y

Which hypothesis do you wish to select?

input>type4

% The consultation begins at this point with the system
% prompting the user for the location of the building.

Where is the building in question located?

units >NIL

input>ottawa

(VARIABLE > LOCATION = OTTAWA)
(VARIABLE > JAN_TEMP_2HAF_PER = -25.0)
(VARIABLE > JAN_TEMP_1_PER = -27.0)
(VARIABLE > JULY_TEMP_2HAF_DRY = 30.0)
(VARIABLE > JULY_TEMP_2HAF_WET = 23.0)
(VARIABLE > DEG_DAYS_<_18 = 4673)
(VARIABLE > GRD_SNOW_LD = 2.9)

(VARIABLE > !15_MIN_RAIN = 23)
 (VARIABLE > !1_DAY_RAIN = 93)
 (VARIABLE > ANN_TOT_PCPN = 846)
 (VARIABLE > !1_IN_10_WIND = 0.37)
 (VARIABLE > !1_IN_30_WIND = 0.37)
 (VARIABLE > !1_IN_100_WIND = 0.37)
 (VARIABLE > SEISMIC_ZONE = 2)
 (VARIABLE > ACCELER_RATIO = 0.04)

Masonry parameters loaded

CISC handbook loaded

(VARIABLE > E_STEEL = 200000)

Please enter a value for the variable:

\$WALL

units > NIL

input>south_wall

(VARIABLE > \$WALL = SOUTH_WALL)

% After the initial parameters have been loaded the

% system begins to query the user about the wall in

% question in an attempt to verify hypothesis 4.

How strongly do you believe that SOUTH_WALL IS CRACKED ?

$0 \leq CF \leq 1$

input>1.0

To what degree is this true: SOUTH_WALL IS AN INFILL ? $0 \leq CF \leq 1$

input>1.0

To what degree do you believe that SOUTH_WALL IS NON
LOAD BEARING ?

$0 \leq CF \leq 1$

input>1.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT FRAME WALL INTERACTION

SHOULD BE INVESTIGATED)
 (RULE GR1 USED)
 (CERTAINTY FACTOR $\Rightarrow$ 1.0)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT SHRINKAGE OF
 SOUTH_WALL SHOULD BE INVESTIGATED)
 (RULE GR1 USED)
 (CERTAINTY FACTOR $\Rightarrow$ 1.0)

How certain are you that CRACKS APPEAR NEAR THE CENTER OF
 THE PANEL ?
 $0 \leq CF \leq 1$

input > 1.0

How strongly do you believe that CRACKS PASS THROUGH
 THE MORTAR ?
 $0 \leq CF \leq 1$

input > 1.0

(THE POSSIBILITY THAT LATERAL LOADING HAS CAUSED CRACKING IN
 THE PANEL SHOULD BE INVESTIGATED)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT SHOULD INVESTIGATE
 LATERAL LOADING)
 (RULE GR11 USED)
 (CERTAINTY FACTOR $\Rightarrow$ 1.0)

% The system confirms that either frame wall interaction or
 % shrinkage should be investigated. Since the user selected
 % hypothesis type 4 frame wall interaction will be investigated.
 % MASON now attempts to narrow the possible types of failure
 % mechanisms.

To what degree is this true: THE CRACK ORIENTATIONS ARE DIAGONAL
 TO THE PANEL ?
 $0 \leq CF \leq 1$

input > 1.0

To what degree do you believe that THERE IS EVIDENCE OF CRUSHING

AT THE CORNERS ?

$0 \leq CF \leq 1$

input>0.60

(THERE IS EVIDENCE TO SUGGEST THAT SHOULD INVESTIGATE FAILURE
TYPE 14A)

(RULE GR13 USED)

(CERTAINTY FACTOR $\Rightarrow$ 0.60)

To what degree is this true: FAILURE IS LOCALIZED TO END PANELS ?

$0 \leq CF \leq 1$

input>1.0

How strongly do you believe that THERE IS SOME EVIDENCE OF
AN EARTHQUAKE ?

$0 \leq CF \leq 1$

input>current rule

ID009

input>show rule id009

rule $\Rightarrow$ ID009

IF

(THERE IS SOME EVIDENCE OF AN EARTHQUAKE)

THEN

(V_LATERAL $\leftarrow$

(ACCELER_RATIO * SEISMIC_RESPONSE * K_EARTHQUAKE *
IMPORTANCE_FACTOR * FOUNDATION_FACTOR * WEIGHT_FRAME))

MESSAGE

(LATERAL LOAD CALCULATED FROM NBCC 14-1-9-1.4 PAGE148)

FUNCTION

(MINNER)

ATTENUATION_FACTOR

(1.0)

T

% User requests a description of the variable V_lateral.

input>describe v_lateral

Calculated lateral load on frame (from NBCC)

T

input>units v_lateral

KN

T

input> 0.75

Please enter a value for the variable:

\$BUILDING

units >NIL

input>eatons_warehouse

(VARIABLE > \$BUILDING = EATONS_WAREHOUSE)

% The following shows the system querying the user for
 % specific information about the building in question
 % so that it will be able to calculate the design
 % earthquake force.

How certain are you that SUPPORTING STRUCTURE OF
 EATONS_WAREHOUSE IS MOMENT RESISTING FRAME ?

$0 \leq CF \leq 1$

input>0.0

What is the overall building height?

units >MM

input>4500.0

(VARIABLE > H_FRAME = 4500.0)

What is the dimension of the building parallel to the applied forces?

units >M

input>28.0

(VARIABLE > D_{||} = 28.0)
 (VARIABLE > T_PERIOD = 0.0765374)
 (VARIABLE > SEISMIC_RESPONSE = 1.0)

How strongly do you believe that SUPPORTING STRUCTURE OF
 EATONS_WAREHOUSE HAS NON STANDARD FRAME ?

$0 \leq CF \leq 1$

input> 1.0

(VARIABLE > K_EARTHQUAKE = 2.0)

To what degree is this true: EATONS_WAREHOUSE IS A POST
 DISASTER BUILDING ?

$0 \leq CF \leq 1$

% The user queries the system in an attempt to find
 % out how it calculated the seismic response factor.
 % The user calculates the variable in order to verify
 % the rule used by MASON

input> how did you get $\Rightarrow$ seismic_response

(SEISMIC_RESPONSE CALCULATED FROM:)
 (THERE IS SOME EVIDENCE OF AN EARTHQUAKE)
 (@\$BINDER SEISMIC_RESPONSE
 (IF (> (/ 0.5 (SQRT T_PERIOD)) 1.0) 1.0 (/ 0.5 (SQRT T_PERIOD))))
 T

input> 0.5 / (SQRT T_PERIOD)

1.8073112

input> variables_list

LOCATION	OTTAWA
E_STEEL	200000
H_FRAME	4500.0
ACCELER_RATIO	0.04
SEISMIC_RESPONSE	1.0
K_EARTHQUAKE	2.0
T_PERIOD	0.0765374
D	28.0
GRD_SNOW_LD	2.9

T

input>repeat query

To what degree do you believe that EATONS_WAREHOUSE IS A
POST DISASTER BUILDING ?

$0 \leq CF \leq 1$

T

input>0.0

(VARIABLE > IMPORTANCE_FACTOR = 1.0)

How certain are you that STRUCTURE FOUNDATIONS ARE ON VERY
LOOSE COARSE SOILS OR VERY SOFT FINE GRAINED SOILS ?

$0 \leq CF \leq 1$

input>current hypothesis

TYPE4

input>elaborate type4

Frame wall interaction failure due to lateral loads.

T

input>hypothesis_list

(TYPE1 TYPE2 TYPE3 TYPE4 TYPE5)

input>elaborate type5

Failure due to shrinkage effects.

T

input>repeat query

How strongly do you believe that STRUCTURE FOUNDATIONS ARE
ON VERY LOOSE COARSE SOILS OR VERY SOFT FINE GRAINED SOILS ?

$0 \leq CF \leq 1$

T

input>0.0

To what degree is this true: STRUCTURE FOUNDATIONS ARE
ON COMPACT COARSE SOILS OR STIFF FINE GRAINED SOILS ?

$0 \leq CF \leq 1$

input>0.67

(VARIABLE > FOUNDATION_FACTOR = 1.3)

What is the dimension of the building perpendicular to the applied forces?

units > M

input>50.0

(VARIABLE > D_l = 50.0)

What is the total DEAD weight of the building?

units > KPA

input>2.5

(VARIABLE > WEIGHT_KPA = 2.5)

(VARIABLE > WEIGHT_FRAME = 4312.0)

(VARIABLE > V_LATERAL = 448.448)

% At this point the user wishes to increase the load on
% the roof of the structure. The user wishes to change
% the variable weight_kpa to 3.0. The help facility is
% used to elicit information on how this could be done.

How many infill panels are there in the direction of the lateral load?

units > NIL

input>help@

The help facility is not working yet!

For more information type (HELP@ TOPICS).

T

input>help@ topics

Current topics are:

ADD_NEW_RULE

ATTENUATION

BOOLEAN_FN
CHANGE
COMPILE
CURRENT
CREATE_RULE_BASE
DESCRIBE
DELETE_A_RULE
FUNCTIONS
GET@
GLOBALS
HAVE
HOW
HYPOTHESIS
MAXER
MENU
MINNER
REPEAT_QUERY
RULES_LIST
SHOW_RULE
STATS_RULE
STATUS
STRENGTHEN
TOPICS
USER_MANUAL
UNITS
VARIABLES
WHY

Do you wish help on a topic? Y/N/TOPIC/?/Q

input>y

Please enter a topic

input>change

(CHANGE variable TO new value) —> A context modifying function.

CHANGE changes the binding of the specified variable to the desired variable. CHANGE initiates a forward chaining procedure that maintains consistency in the context. Be aware the procedure may take some time.

└

USAGE input> (CHANGE JUNK TO 4.3)

T

% User requests a list of all bound variables and
% their respective bindings.

input> variables_list

LOCATION	OTTAWA
E_STEEL	200000
H_FRAME	4500.0
V_LATERAL	448.48
ACCELER_RATIO	0.04
SEISMIC_RESPONSE	1.0
K_EARTHQUAKE	2.0
IMPORTANCE_FACTOR	1.0
FOUNDATION_FACTOR	1.3
WEIGHT_FRAME	4312.0
T_PERIOD	0.0765374
D	28.0
D _⊥	50.0
WEIGHT_KPA	2.5
GRD_SNOW_LD	2.9

T

% Here the desired change is made and the results of
% that change are subsequently propagated forward.
% Variables whose values are changed are rebound.

input> change weight_kpa to 3.0

(VARIABLE > WEIGHT_KPA = 3.0)

(VARIABLE > WEIGHT_FRAME = 5012.0)

(VARIABLE > V_LATERAL = 521.248)

T

% The consultation process is continued. The system begins
% to request specific information about the masonry walls
% in order to determine whether or not allowable stresses
% were exceeded.

input> repeat query

How many infill panels are there in the direction of the lateral load?

units > NIL

T

input > 2

(VARIABLE > NO_BAYS = 2)

(VARIABLE > V_PANEL = 130.312)

What is the length of a typical bay?

units > MM

input > 7000.0

(VARIABLE > L_FRAME = 7000.0)

(VARIABLE > ALPHA = 0.2102946)

Are the units brick or block?

input > block

(VARIABLE > \$UNITS = BLOCK)

Could you supply the type of blocks used? Y/N

Choices are:

(HOLLOW 175_SOLID SOLID_PIER)

input > y

input > hollow

(VARIABLE > FACE_SHELL_THICKNESS = 32.0)

(VARIABLE > WIDTH_UNIT = 190.0)

(VARIABLE > EQ_WIDTH_UNIT = 106.0)

(VARIABLE > WALL_MASS = 223.0)

(VARIABLE > WALL_MASS_LT = 181.0)

(VARIABLE > *PROPS* = T)

(VARIABLE > F_BEARING_1 = 1.2990859)

(VARIABLE > F_BEARING_2 = 0.3387008)

(VARIABLE > F_BEARING_MAX = 1.2990859)

Is the mortar type N, M or S?

input > n

(VARIABLE > \$MORTAR = N)

Could you provide an estimate of the compressive strength of the units? Y/N

input>y

input>10.0

(VARIABLE > FB_PRIME = 10.0)

(VARIABLE > FM_PRIME = 5.4)

(TAKEN FROM CSA S304-M78 !4-3-3-7)

(VARIABLE > SIGMA_ALLOW = 1.08)

% It has been determined that there is some evidence to
 % suggest that the crushing in the corners of the wall
 % was caused by excessive lateral loads. The system now
 % proceeds to determine the cause of the diagonal cracks.

(THERE IS EVIDENCE TO SUGGEST THAT SHOULD INVESTIGATE FAILURE
 TYPE 4)

(RULE ID420 USED)

(CERTAINTY FACTOR $\Rightarrow$ 0.60)

How strongly do you believe that SUPPORTING STRUCTURE OF
 EATONS_WAREHOUSE IS PINNED FRAME ?

$0 \leq CF \leq 1$

input>1.0

To what degree is this true: SUPPORTING STRUCTURE OF
 EATONS_WAREHOUSE IS BRACED ?

$0 \leq CF \leq 1$

input>1.0

What is the area of bracing in the frame?

units >MM

input>600.0

(VARIABLE > AREA_BRACE = 600.0)

To what degree do you believe that SUPPORTING STRUCTURE OF
EATONS_WAREHOUSE IS CONCRETE ?

$0 \leq CF \leq 1$

input>0.0

How certain are you that SUPPORTING STRUCTURE OF
EATONS_WAREHOUSE IS STEEL ?

$0 \leq CF \leq 1$

input>1.0

(VARIABLE > E_FRAME = 200000)

(VARIABLE > K_FRAME = 10203.4652042)

Please provide the number of stories in the building
units >NIL

input>1

(VARIABLE > NO_STORIES = 1)

(VARIABLE > V_ALLOW_STR = 0.16)

(ALLOWABLE SHEAR IN CONCRETE BLOCK TAKEN CSAS304-M78 TABLE 5 P35)

(VARIABLE > V_ALLOW = 118.7199999)

(VARIABLE > G_WALL = 2160.0)

(G_WALL HAS BEEN CHOSEN AS THE LEAST OF !400*FM_PRIME OR 8000 MPA)

(VARIABLE > DELTA_FRAME = 0.3999999)

(VARIABLE > V_FRAME = 4.0813854)

(VARIABLE > V_WALL = 126.2306145)

(THERE IS EVIDENCE TO SUGGEST THAT SHEAR IN SOUTH_WALL EXCEEDS
SHEAR CAPACITY OF WALL)

(RULE ID402 USED)

(CERTAINTY FACTOR $\Rightarrow$ 0.60)

(THERE IS EVIDENCE TO SUGGEST CONCLUDE FAILURE TYPE 4)

(RULE ID401 USED)

(CERTAINTY FACTOR $\Rightarrow$ 0.60)

% The system concludes that failure was caused
 % by lateral loading due to an earthquake. The measure
 % of belief (0.6) was calculated simply by taking the
 % lowest measure of certainty of the supporting evidence.

Do you wish another run? Y/N

input>n

Ok, session over

T

3 lisp>

MASON concluded that cracking in the end panels was likely due to extreme lateral loading exceeding the shear capacity of the masonry walls. The infill panels at the ends, although not designed to carry lateral, in fact took on a substantial portion of the earthquake loads causing them to crush at the corners and split near the center.

6.4 Sample Session Three

The building considered in the third sample session consisted of a low rise masonry structure with load bearing exterior walls. The building was rectangular in plan and cracks were noticed along the north face of the building. A preliminary investigation detected that these cracks occurred near the window openings of the wall. The cracks were oriented in the vertical direction and passed only through the mortar. The building was relatively new, less than one year old, when cracking occurred. The ends of the north face wall were restrained by the exterior walls running perpendicular to that wall. The general layout for this case is shown in Figure 6.3.

The following information was available during the consultation with MASON.

Figure 6-3: Case for sample session three.

Location = Montreal

Name of Building = Everest Inc.

$f'_b = 10.0 \text{ Mpa}$

Mortar = Type N

Grouted = No

Aggregate Type = Cinder

Condition Laid = Saturated

Curing = High Temperature

$H_{wall} = 5000.0mm$

$L_{wall} = 15000.0mm$

Window height = 500.0mm

Bond Pattern. = Running Bond

$K_{connection} = 45000.0N/mm$

3 lisp> (restart)

% The user restarts the MASON system from the
 % previous consultation in order to investigate
 % cracking in another building. During this session
 % the user does not have any conjectures as to the
 % the possible failure mechanism so MASON chooses a
 % possible failure mode and proceeds to investigate it.

Hi...

When you wish to initiate a run type RUN.

input>run

Do you wish to enter any initial facts? Y/N

input>n

Do you wish to select a hypothesis? Y/N

input>n

Ok, I will pick one for you.

TYPE1 has been chosen.

Where is the building in question located?

units >NIL

input>montreal

(VARIABLE > LOCATION = MONTREAL)

(VARIABLE > JAN_TEMP_2HAF_PER = -23.0)

(VARIABLE > JAN_TEMP_1_PER = -26.0)

```

(VARIABLE > JULY_TEMP_2HAF_DRY = 30.0)
(VARIABLE > JULY_TEMP_2HAF_WET = 23.0)
(VARIABLE > DEG_DAYS_<_18 = 4471)
(VARIABLE > GRD_SNOW_LD = 2.7)
(VARIABLE > !15_MIN_RAIN = 23)
(VARIABLE > !1_DAY_RAIN = 90)
(VARIABLE > ANN_TOT_PCPN = 999)
(VARIABLE > !1_IN_10_WIND = 0.3699999)
(VARIABLE > !1_IN_30_WIND = 0.3699999)
(VARIABLE > !1_IN_100_WIND = 0.3699999)
(VARIABLE > SEISMIC_ZONE = 2)
(VARIABLE > ACCELER_RATIO = 0.04)
(VARIABLE > *FILES* = 'T')

```

Masonry parameters loaded

CISC handbook loaded

```
(VARIABLE > E_STEEL = 200000)
```

Please enter a value for the variable: *

\$WALL

units > NIL

input> north_face

```
(VARIABLE > $WALL = NORTH_FACE)
```

To what degree is this true: NORTH_FACE IS CRACKED ? $0 \leq CF \leq 1$

input> 1.0

To what degree do you believe that NORTH_FACE IS AN INFILL ?

$0 \leq CF \leq 1$

```

% Here the user indicates that the wall is not an infill
% therefore failure type 1 cannot be established as it
% requires that the wall be an infill wall.

```

input> 0.0

(HYPOTHESIS TYPE1 NOT CONFIRMED)

% The user wishes another run. MASON will retain

% all the information entered until this point and
% investigate another hypothesis.

Do you wish another run? Y/N

input> y

Do you wish to enter any initial facts? Y/N

input> n

Do you wish to select a hypothesis? Y/N

input> n

Ok. I will pick one for you.
TYPE2 has been chosen.

(HYPOTHESIS TYPE2 NOT CONFIRMED)

% Failure type 2 cannot be established because it also
% requires that the wall be an ipfill panel.

Do you wish another run? Y/N

input> y

Do you wish to enter any initial facts? Y/N

input> n

Do you wish to select a hypothesis? Y/N

input> n

% The system picks the third hypothesis in an attempt to
% diagnose the cause of cracking.

Ok. I will pick one for you.
TYPE3 has been chosen.

(HYPOTHESIS TYPE3 NOT CONFIRMED)

% Again the hypothesis cannot be confirmed because the wall

% is not an infill.

Do you wish another run? Y/N

input>y

Do you wish to enter any initial facts? Y/N

input>n

% The user decides to select a hypothesis in an attempt
% to determine the cause of cracking. Type 5. failure
% due to shrinkage effects is chosen.

Do you wish to select a hypothesis? Y/N

input>y

Which hypothesis do you wish to select?

input>?

(TYPE1 TYPE2 TYPE3 TYPE4 TYPE5)

For a description type
ELABORATE hypothesis
or Q to quit

Which hypothesis do you wish to select?

input>type5

Please enter a value for the variable:

\$BUILDING

units >NIL

input>everest_inc

(VARIABLE > \$BUILDING = EVEREST_INC)

How certain are you that NORTH_FACE IS NON LOAD BEARING ?

$0 \leq CF \leq 1$

% The following shows the user examining a rule, the status
% of the rule and the available statistics on the rule's usage.

input>current rule

GR16

input>menu

MENU

commands are;

ADD_NEW_RULE

ADD_HYPOTHESIS

CHANGE variable TO new value

COMPILE desired rule base

CURRENT rule or hypothesis

CREATE_RULE_BASE

DELETE_A_RULE

DELETE_A_HYPOTHESIS

DESCRIBE variable

ELABORATE hypothesis

Type C to continue menu or X to exit menu.

input>c

commands are;

GET@ KNOWN FACTS

GET@ RULES USED

GET@ VALUE variable

HAVE [you] USED RULE rule identifier

HELP@ [desired topic]

HOW DID [you get] $\Rightarrow$ some conclusion

HOW COULD [you get] $\Rightarrow$ some conclusion

HYPOTHESIS_LIST

MENU

REPEAT QUERY

RESET

Type C to continue menu or X to exit menu.

input>x

input>have you used rule gr16

No. would you like to examine it? Y/N

input>y

rule ⇒GR16

IF

(\$WALL IS CRACKED)

(~\$WALL IS AN INFILL)

(~\$WALL IS NON LOAD BEARING)

THEN

(SHRINKAGE OF \$WALL SHOULD BE INVESTIGATED)

MESSAGE

NIL

FUNCTION

(MINNER)

ATTENUATION_FACTOR

(1.0)

T

input>menu

MENU

commands are:

RULES_LIST

SHOW [RULE rule identifier]

STATS RULE rule identifier

STATUS OF rule ⇒ rule identifier

STATUS OF fact ⇒ a certain fact

STATUS OF variable ⇒ variable name

UNITS variable

VARIABLES_LIST

WHY DID [you need] ⇒ some fact or variable

WHY DO [you need] ⇒ some fact or variable

Q to quit to top level.

? to receives instructions at a given level

Type C to continue menu or X to exit menu.

input>x

input>status of rule gr16

UNUSED

input>stats rule gr16

rule $\Rightarrow$ GR16

AUTHOR

CORNICK

DATE_LAST_MODIFIED

16-MAY-1985

TIMES_USED

1

TIMES_CONSIDERED

3

PERCENTAGE_USE

33.3333333

T

input>repeat query

To what degree is this true: NORTH_FACE IS NON LOAD BEARING ?

$0 \leq CF \leq 1$

T

input>0.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT SHRINKAGE OF NORTH_FACE
SHOULD BE INVESTIGATED)

(RULE GR16 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

To what degree do you believe that CRACKS ARE ORIENTED IN
THE VERTICAL DIRECTION ?

$0 \leq CF \leq 1$

input>status of rule $\Rightarrow$ gr16

FIRED

input>stats rule gr16

rule $\Rightarrow$ GR16

AUTHOR

CORNICK
DATE_LAST_MODIFIED
16-MAY-1985
TIMES_USED
2
TIMES_CONSIDERED
3
PERCENTAGE_USE
66.666666
T

% The user then proceeds with consultation that eventually
% leads to the system suggesting that failure was caused by
% shrinkage effects.
input>repeat query

How certain are you that CRACKS ARE ORIENTED IN THE
VERTICAL DIRECTION ?
 $0 \leq CF \leq 1$
T

input>1.0

How certain are you that EVEREST_INC IS MORE THAN ONE YEAR OLD ?
 $0 \leq CF \leq 1$

input>0.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT POSSIBLE CAUSE OF
CRACKING IS UNIT SHRINKAGE)
(RULE GR14 USED)
(CERTAINTY FACTOR $\Rightarrow$ 1.0)

What is the height of the wall?
units >MM

input>5000.0

(VARIABLE > HEIGHT_WALL = 5000.0)

How long is the wall?
units >MM

input>15000

(VARIABLE > LENGTH_WALL = 15000)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT NORTH_FACE IS
A LONG WALL)

(RULE ID031 USED)

(CERTAINTY FACTOR $\Rightarrow$ 1.0)

How strongly do you believe that CRACKS PASS THROUGH
THE MORTAR ?

$0 \leq CF \leq 1$

input>how did you get $\Rightarrow$ north_face is a long wall

(NORTH_FACE IS A LONG WALL DEMONSTRATED BY)

(0.5 $\leftarrow$ (/ HEIGHT_WALL LENGTH_WALL) 0.5)

T

input>why did you need $\Rightarrow$ north_face is a long wall

(NORTH_FACE IS A LONG WALL IS NEEDED TO SHOW:)

(SHOULD INVESTIGATE FAILURE TYPE 5)

T

input>how could you get $\Rightarrow$ north_face is a long wall

(NORTH_FACE IS A LONG WALL HAS BEEN ESTABLISHED)

T

input>how could you get $\Rightarrow$ should investigate failure type 5

(SHOULD INVESTIGATE FAILURE TYPE 5 CAN BE ESTABLISHED BY SHOWING)

(POSSIBLE CAUSE OF CRACKING IS UNIT SHRINKAGE) KNOWN

(NORTH_FACE IS A LONG WALL) KNOWN

(CRACKS PASS THROUGH THE MORTAR)

T

input>repeat query

To what degree is this true: CRACKS PASS THROUGH THE MORTAR ?

$0 \leq CF \leq 1$

T

input>1.0

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT SHOULD INVESTIGATE
FAILURE TYPE 5)
(RULE GR15 USED)
(CERTAINTY FACTOR $\Rightarrow$ 1.0)

To what degree do you believe that NORTH_FACE HAS OPENINGS ?
 $0 \leq CF \leq 1$

input>1.0

How certain are you that CRACKS APPEAR NEAR THE OPENINGS
OF NORTH_FACE ?
 $0 \leq CF \leq 1$

input>1.0

How strongly do you believe that THERE ARE MOVEMENT OR
EXPANSION JOINTS IN NORTH_FACE ?
 $0 \leq CF \leq 1$

input>0.0

(VARIABLE > L_WALL = 15000)

Are the units brick or block?

input>block

(VARIABLE > UNITS = BLOCK)

To what degree is this true: BLOCK HAS BEEN HIGH PRESSURE
STEAM CURED ?
 $0 \leq CF \leq 1$

input>how did you get $\Rightarrow$ L_wall

(L_WALL CALCULATED FROM:)
(SHOULD INVESTIGATE FAILURE TYPE 5)
(THERE ARE MOVEMENT OR EXPANSION JOINTS IN NORTH_FACE)

(0\$^BINDER 'L_WALL LENGTH_WALL)

T

input>how could you get $\Rightarrow$ L_wall

(L_WALL CAN BE ESTABLISHED BY SHOWING)

(SHOULD INVESTIGATE FAILURE TYPE 5) KNOWN

(THERE ARE MOVEMENT OR EXPANSION JOINTS IN NORTH_FACE) KNOWN

OR

(L_WALL CAN BE ESTABLISHED BY SHOWING)

(SHOULD INVESTIGATE FAILURE TYPE 5) KNOWN

(THERE ARE MOVEMENT OR EXPANSION JOINTS IN NORTH_FACE) KNOWN

(0\$^> (GET_VAL SPACING) 6.0)

T

input>0.0

To what degree is this true: BLOCK WAS SATURATED WHEN LAID ?

$0 \leq CF \leq 1$

input>1.0

Can you provide the type of aggregate used for the blocks? Y/N

input>y

Choices are SG (sand gravel), EX (expanded shale).

C (cinder) and ES (expanded slag).

input>c

(VARIABLE > \$AGGREGATE = C)

(VARIABLE > EPSILON_SH_WALL = 0.069)

(VARIABLE > DELTA_WALL = 10.3499999)

(DEFLECTION OF WALL CALCULATED AS LENGTH TIMES SHRINKAGE
VALUE FOR ASSEMBLAGE)

Is the mortar type N, M or S?

input>n

(VARIABLE > \$MORTAR = N)

Could you provide an estimate of the compressive strength of the units? Y/N

input>y

input>10.0

(VARIABLE > FB_PRIME = 10.0)

(VARIABLE > FM_PRIME = 5.4)

(TAKEN FROM CSA S304-M78 14-3-3-7)

(VARIABLE > E_WALL = 5400.0)

(E_WALL HAS BEEN CHOSEN AS THE LEAST OF $1000 \cdot FM_PRIME$ OR 20000 MPA)

What is the height of the opening near the cracks?

units >MM

input>500.0

(VARIABLE > HEIGHT_OPENING = 500.0)

(VARIABLE > H_WALL = 4500.0)

Could you supply the type of blocks used? Y/N

Choices are:

(HOLLOW 175_SOLID SOLID_PIER)

input>n

All right I will assume Hollow Block

(VARIABLE > FACE_SHELL_THICKNESS = 32.0)

(VARIABLE > WIDTH_UNIT = 190.0)

(VARIABLE > EQ_WIDTH_UNIT = 106.0)

(VARIABLE > WALL_MASS = 223.0)

(VARIABLE > WALL_MASS_LT = 181.0)

(VARIABLE > *PROPS* = T)

(VARIABLE > AREA_WALL = 288000.0)

Please estimate the stiffness of the end connections of the wall
units > N/MM

input>45000.0

(VARIABLE > K_CONNECT = 45000.0)
(VARIABLE > SIGMA_WALL = 0.6644079)
(STRESS IN WALL EQUALS LENGTH TIMES SHRINKAGE DIVIDED BY AREA TIMES
MODULUS OF ELASTICITY)

To what degree do you believe that NORTH_FACE HAS
BEEN GROUTED ?

$0 \leq CF \leq 1$

input>how did you get $\Rightarrow$ sigma_wall

(SIGMA_WALL CALCULATED FROM:
(SHOULD INVESTIGATE FAILURE TYPE 5)
(0.5 * BINDER * SIGMA_WALL
(/ (* DELTA_WALL E_WALL)
(1+ L_WALL (/ (* (* 2 AREA_WALL) E_WALL) K_CONNECT))))

T

input>0.0

How certain are you that BOND PATTERN IS RUNNING BOND ?

$0 \leq CF \leq 1$

input>1.0

(VARIABLE > SIGMA_ALLOW = 0.22)
(TAKEN FROM TABLE 7 P36 CSA304)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT TENSION STRESSES
IN WALL EXCEED ALLOWABLE STRESS)

(RULE ID505 USED)
(CERTAINTY FACTOR $\Rightarrow$ 1.0)

(THERE IS DEFINITE EVIDENCE TO SUGGEST THAT OPENINGS WERE
IMPROPERLY DETAILED)

(RULE ID505 USED)
(CERTAINTY FACTOR $\Rightarrow$ 1.0)

(THERE IS DEFINITE EVIDENCE TO SUGGEST CONCLUDE FAILURE
TYPE 5)
(RULE ID501 USED)
(CERTAINTY FACTOR $\Rightarrow$ 1.0)

Do you wish another run? Y/N

input>n
Ok session is over.
Goodbye ...
T

4 lisp>

The probable cause of failure, determined by MASON, was that cracking was caused by the shrinkage of the masonry units in the wall. Tensile stresses were developed in the wall due to the restraining action of the walls running perpendicular to the north face wall. The tensile stresses were increased due to the presence of openings in the wall which reduced the net wall area. MASON suggested that cracking was caused by improper detailing around the wall openings.

6.5 Summary

This chapter contained three sample interactions, with MASON, that were intended to demonstrate the operation and capabilities of the system. A preliminary field inspection was assumed to have occurred before each consultation. The examples demonstrated the feasibility of such a system. The next stage of development for MASON would be the application of the models to actual case studies in an attempt to refine the current models.

Chapter 7

Conclusions and Recommendations

As we succeed in broadening and deepening our knowledge— theoretical and empirical—about computers, we shall discover that in large part of their behavior is governed by simple general laws, that what appeared as complexity in the computer program was to a considerable extent complexity of the environment to which the program was seeking to adapt its behavior.

—H. A. Simon

The Sciences of the Artificial

7.1 Introduction

This chapter contains conclusions and recommendations that were based on this research. The conclusions were made with respect to the original objectives outlined in Chapter 1. The limitations of the EMASON and MASON systems are also discussed. Recommendations for the continuance of this work is presented in the last part of this chapter.

7.2 Conclusions

The first major objective of this thesis, briefly stated, was the investigation of the applicability of AI tools and techniques to Civil Engineering problems. The most promising area of AI research with respect to Engineering is the field of Knowledge Based Expert Systems. Based on the Knowledge Paradigm, systems of this type appear to be considerably more flexible and robust than programs implemented using traditional programming techniques. This is due to the separation of domain knowledge and control knowledge. Changes to a system of this type can be confined to system's knowledge base. This property of KBES was noticed during the implementation of the MASON system.

In implementing the EMASON system two key issues were addressed: 1) selection of a problem solving strategy and 2) the choice of the type of knowledge representation. There are several problem solving strategies that can be applied successfully to engineering tasks, however no overall strategy exists that could encompass the domain of Civil Engineering. None of the solution techniques outlined in Chapter 3 could be applied in isolation to

moderately complex problems—thus the choice of an appropriate solution technique is dependent upon the nature of the domain and task at hand. A thorough investigation of the domain and the problem should be made before a problem solving strategy is chosen. Furthermore, moderately complex problems, such as preliminary structural design, would require a hybrid strategy, incorporating the advantages of several solution techniques.

The selection of Knowledge Representation techniques was found to be dependent on the Problem Solving Strategy used, as well as on the nature of the problem and domain. The choice of one will effect the other. Two techniques for representing knowledge were investigated, namely production rules and a frame based representation language. Production rules are a procedural type of knowledge representation, while frames are a hybrid type of representation, combining procedures and declarations. Production rules are the easiest to comprehend and implement. Expert knowledge can be encoded very easily into a production rule format.

Frames (or frame based representations) appear to be particularly appropriate for use in engineering applications. Frame based representations allow the user to construct a structured relational network between entities. Generic objects can be created and the information contained in these can be inherited by other objects. Frames are particularly useful in providing the user with a simple means of specifying default values and procedures to be executed if certain conditions are met. Frames provide an excellent underlying representation for the description of building structures.

A purely declarative representation, Programming Logic, was investigated on a cursory level using PROLOG, a language that uses predicate logic. On the basis of very limited experience with PROLOG, no definite conclusions could be made as to the appropriateness of PROLOG for engineering applications. PROLOG does, however, warrant further study.

The second objective was the development of tools that would serve as a starting point for future research into the application of AI tools and techniques to Civil Engineering problems. A rule based system with a fairly standard inference engine was implemented to allow for the development of engineering applications programs that would be difficult to implement using traditional techniques. The inference engine differed from others in its ability to handle numerical computations. EMASON used the backward chaining mechanism to evaluate mathematical expressions and EMASON provided for the free mixing of

symbolic and numerical data within the rules. In order to provide flexibility to the user, EMASON allowed the user to modify the context directly during execution. The user was restricted to modifying only those parameters that were not derived from other data. The context modification procedure was found to be rather time consuming but necessary.

The development of the EMASON ES framework stopped when an operational version of the system had been implemented. EMASON was intended as an experimental tool to determine the applicability and requirements of such a rule based system to engineering problems. The successful handling of mathematical expressions, the ability for the user to modify parameters and the context, and the ease with which information from codes, specifications, textbooks, and experts was encoded into production rules demonstrates the applicability of EMASON. The EMASON system should be considered as a high level programming language in which the user encodes knowledge as a collection of production rules. Although the ability to exit the EMASON system to access conventional programs or data base systems was not implemented this would be relatively easy to do using the system calls provided in PSL. This feature would enhance the power and flexibility of the system.

A rudimentary frame based representation language was also implemented and successfully investigated. Although EMASON does not currently use this language, future implementations will. Experience with frames, particularly in the area of building representation, was encouraging. The ability of frames to represent hierarchical or almost hierarchical systems makes them appropriate for describing building structures.

MASON was developed to demonstrate the application of EMASON to a Civil Engineering problem (in fact the two systems were developed concurrently). A backward chaining approach was found to be suitable for the determination of failure modes in walls since the problem was a diagnostic one and it was possible to categorize the failure mechanisms. MASON is currently in an initial phase of development. The models that comprise the knowledge base are not truly representative of actual failure mechanisms in masonry walls. The MASON knowledge base should be considered as the first draft of a future, more comprehensive, knowledge base. The development of MASON and EMASON took approximately six months, with the majority of the effort going into the implementation of the EMASON framework. Currently MASON consists of 172 rules, 101 variables, and five failure models incorporated into the rule base. The large part of future efforts will be devoted to the acquiring of knowledge for the MASON knowledge base.

During the knowledge acquisition phase it was found that certain rules could be collected together to form a class of rules that deal with the same area of knowledge. The masonry domain was found to be a subset of the broader domain of building design and construction, which contains several different classes of rules. Many of the rules encoded were rules that contained knowledge of a general nature concerning building design and construction. It should be possible to use these rules in other systems intended to operate in the structural engineering domain.

7.3 Limitations of EMASON/MASON

This section discusses the limitations of the current implementations of the EMASON and MASON systems. The drawbacks reflect some of the limitations in the current state of the art in KBES technology.

The first and perhaps most serious limitation of EMASON/MASON is that it does not degrade well at the boundaries of its knowledge. Human experts degrade gracefully when they reach the limits of their knowledge; they can recognize the boundaries of their knowledge and are able to draw on other resources. They are able to reason by analogy or to break rules in order to solve problems. Unlike humans, EMASON/MASON does not know the limits of its expertise. There exists a very well defined boundary beyond which the system fails to arrive at a solution. The system is not able to recognize analogous situations or break its own rules. This limitation makes MASON and systems developed using EMASON very narrow in their scope and unable to function in multiple domains.

Another serious drawback of EMASON/MASON is the high cost of acquiring knowledge. The task of acquiring knowledge is difficult and requires considerably more time and effort than the implementation of the Expert System framework. It has been estimated that a KBES takes approximately five man years to develop. Most of this time is taken up in the development of the knowledge base. The lack of well developed Knowledge Acquisition tools in EMASON/MASON has several consequences:

- 1) Knowledge must be hand coded into a production rule format requiring considerable time and effort by the Knowledge Engineer and Expert.
- 2) The lack of browsers and editors for the Knowledge Base makes the task of modifying and testing various models and rules in the Knowledge Base difficult.

- 3) The lack of tools to maintain consistency and integrity of the Knowledge Base allows the user to encode rules that could potentially create faults in the Knowledge Base either by generating incorrect solutions or by excluding certain valid solution paths or by creating loops in the Knowledge Base.
- 4) The lack of a Natural Language Interface requires that the Knowledge Engineer or Expert have a good working knowledge of the EMASON framework and the LISP programming language.

It must be noted though, that even with good tools, knowledge acquisition is still very time-consuming.

EMASON/MASON is limited in the type of problems to which it can be applied. The system is unable to solve problems that are based on structure or function. Problems such as preliminary design or planning are very difficult to solve using the production rule format and may require a prohibitive number of rules in the Knowledge Base.

The current implementation of EMASON/MASON has inadequate explanation facilities and is therefore not able to adequately explain its inferences. Explanation is achieved by recreating the inference paths. Explanations are merely the repetition of production rules. The lack of proper explanation is due to the fact the system is based on empirical regularities and not on understanding structure and function.

A less serious limitation is that EMASON/MASON is unable to generate knowledge. The system is not able to create new rules or new models. This would be a particularly desirable feature, however this is beyond the scope of most KBES.

7.4 Recommendations

After implementing and experimenting with the EMASON system the following recommendations are made:

- 1) The development of the EMASON system should be continued in two specific areas: a) An acceptable manner of handling uncertainty should be developed (A rigorous probabilistic approach should be considered), and b) More knowledge acquisition tools should be developed in order to facilitate the development of other applications programs. A rule editor and browsers for the knowledge base should be implemented.

- 2) Frames should be incorporated into the next version of EMASON. Hypothesis, goals, and tasks would be represented as template frames. A hypothesis would be considered satisfied when the appropriate slots in the hypothesis template were filled. The templates would be augmented by production rules that would be used to fill the template slots.
- 3) A constraint management system should be incorporated into EMASON and the system extended so that the user is able to modify derived data.
- 4) A system similar to EMASON should be implemented in PROLOG.

The following recommendations are based on the experience gained while implementing the MASON system. They are;

- 1) The initial five failure modes incorporated into the MASON system should be further developed. A concentrated consultation process should be started in order to elicit expert knowledge in the field of masonry. Initially a few well defined models should be thoroughly explored.
- 2) A field program should be initiated concurrently with the first recommendation to evaluate the failure models as they are developed.
- 3) Failure Type 6, cladding distress, should be implemented. This model would determine, from field examinations of the wall: a) the probable causes of the cladding distress, and b) the input parameters for the VENEER program developed by G. A. Fenton [23]. The VENEER program would be used to corroborate MASON's conclusions.
- 4) The Masonry and Property class rules should be further developed in order to provide a knowledge base that could be used for problems in the structural engineering domain. A new class of rules, Building Design and Construction Rules, should be created. Currently rules of this type reside in the Masonry Class of rules.
- 5) Codes, specifications, and handbooks, such as CSA A23.3-M77, CSA S304-M78, and National Building Code, should be encoded into a rule based format and reside in separate knowledge modules. This would enable modifications to be easily made as these documents are updated.
- 6) The MASON system should be interfaced with data and programs such as those provided by the Canadian Institute of Steel Construction.

Appendix A

Production Rules for Slab Design

```
(RULE ID1
  (IF ((C$` > (/ (GET_VAL B) (GET_VAL D)) 1.1000008) NIL))
  (THEN ((ONE-WAY ACTION IS VALID) (ID1)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID2
  (IF ((C$` <= (/ (GET_VAL B) (GET_VAL D)) 1.5) NIL))
  (THEN ((TWO-WAY ACTION IS VALID) (ID2)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID3 (IF ((ONE-WAY ACTION IS VALID) (ID1)))
  (THEN ((0 EDGE SUPPORT IS VALID) (ID3))
    ((2 EDGE SUPPORT IS VALID) (ID3)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID4 (IF ((TWO-ACTION IS VALID) NIL))
  (THEN ((0 EDGE SUPPORT IS VALID) (ID4))
    ((4 EDGE SUPPORT IS VALID) (ID4)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID5 (IF ((C$` <= (GET_VAL SPAN) 4.0) NIL))
  (THEN ((STEEL DECK IS VALID) (ID5)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID6 (IF ((C$` <= (GET_VAL SPAN) 5.0) NIL))
  (THEN ((PREFAB PANELS IS VALID) (ID6)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID7
  (IF ((C$` AND (>= (GET_VAL SPAN) 3.0)
    (<= (GET_VAL SPAN) 8.0)) NIL))
  (THEN ((REINF SLAB IS VALID) (ID7)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```
(RULE ID8
  (IF ((C$` AND (>= (GET_VAL SPAN) 6.0)
    (<= (GET_VAL SPAN) 13.0)) NIL))
  (THEN ((PRESTR SLAB IS VALID) (ID8)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))
```

```

(RULE ID9
  (IF ((ONE-WAY ACTION IS VALID) (ID1))
    ((2 EDGE SUPPORT IS VALID) (ID3))
    ((STEEL DECK IS VALID) (ID5)))
  (THEN ((STEEL DECK ONE-WAY) (ID9)))
  (MESSAGE_TEXT
    (A POSSIBLE FLOOR SYSTEM IS; ONE WAY
     STEEL DECK SUPPORTED ON TWO EDGES
     PERPENDICULAR TO THE BENDING ACTION))
  (BOOLEAN_FN) (1.0))

(RULE ID10
  (IF ((ONE-WAY ACTION IS VALID) (ID1))
    ((2 EDGE SUPPORT IS VALID) (ID3))
    ((PREFAB PANELS IS VALID) (ID6)))
  (THEN ((PREFAB PANELS ONE-WAY) (ID10)))
  (MESSAGE_TEXT
    (A POSSIBLE FLOOR SYSTEM IS; USE PREFABRICATED
     PRESTRESSED CONCRETE PANELS IN ONE WAY BENDING
     WITH SUPPORT ON TWO EDGES PERPENDICULAR TO THE
     BENDING ACTION))
  (BOOLEAN_FN) (1.0))

(RULE ID11
  (IF ((ONE-WAY ACTION IS VALID) (ID1))
    ((REINF SLAB IS VALID) (ID7)))
  (THEN ((REINF SLAB ONE-WAY) (ID11)))
  (MESSAGE_TEXT
    (A POSSIBLE FLOOR SYSTEM IS; ONE WAY REINFORCED
     CONCRETE SLAB SUPPORTED EITHER ON ZERO OR TWO
     EDGES PERPENDICULAR TO BENDING ACTION))
  (BOOLEAN_FN) (1.0))

(RULE ID12
  (IF ((ONE-WAY ACTION IS VALID) (ID1))
    / ((PRESTR SLAB IS VALID) (ID8)))
  (THEN ((PRESTR SLAB ONE-WAY) (ID12)))
  (MESSAGE_TEXT
    (A POSSIBLE FLOOR SYSTEM IS; USE ONE WAY PRESTRESSED
     CONCRETE SLAB WITH EITHER ZERO OR TWO SUPPRIED EDGES))
  (BOOLEAN_FN) (1.0))

(RULE ID13
  (IF ((TWO-WAY ACTION IS VALID) (ID2))
    ((PRESTR SLAB IS VALID) (ID8)))
  (THEN ((PRESTR SLAB TWO-WAY) (ID13)))
  (MESSAGE_TEXT
    (A POSSIBLE FLOOR SYSTEM IS; USE A TWO WAY PRESTRESSED

```

CONCRETE SLAB SUPPORTED ON EITHER ZERO TWO OR
FOUR EDGES))
(BOOLEAN_FN) (1.0))

(RULE ID14
(IF ((TWO-WAY ACTION IS VALID) (ID2))
((REINF SLAB IS VALID) (ID7)))
(THEN ((REINF SLAB TWO-WAY) (ID14)))
(MESSAGE_TEXT
(A POSSIBLE FLOOR SYSTEM IS; USE A TWO WAY REINFORCED
CONCRETE SLAB SUPPORTED ON EITHER ZERO OR TWO OR FOUR
EDGES))
(BOOLEAN_FN) (1.0))

(RULE ID15
(IF ((@# = *CNTXT_SLAB* 'ONE-WAY') NIL))
(THEN ((@# BINDER 'SPAN (GET_VAL D)) (ID15)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE ID16 (IF ((@# = *CNTXT_SLAB* 'TWO-WAY) NIL))
(THEN ((@# BINDER 'SPAN (GET_VAL B)) (ID16)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

Appendix B

PROLOG Source Code for Slab Design

% FACTBASE:

```

action_type(one_way, 1.2, 20.0).
action_type(two_way, 0.0, 1.5).
material_type(steel_deck, 0.0, 4.0).
material_type(prefab_panels, 0.0, 5.0).
material_type(reinf_slab, 3.0, 8.0).
material_type(prestr_slab, 6.0, 13.0).
material_action(one_way, steel_deck).
material_action(one_way, prefab_panels).
material_action(one_way, reinf_slab).
material_action(one_way, prestr_slab).
material_action(two_way, reinf_slab).
material_action(two_way, prestr_slab).
support_type(reinf_slab, 0).
support_type(reinf_slab, 2).
support_type(reinf_slab, 4).
support_type(steel_deck, 2).
support_type(prefab_panels, 2).
support_type(prestr_slab, 0).
support_type(prestr_slab, 2).
support_type(prestr_slab, 4).
support_allowable(one_way, 0).
support_allowable(one_way, 2).
support_allowable(two_way, 0).
support_allowable(two_way, 2).
support_allowable(two_way, 4).

```

% RULEBASE:

```

slab_parameter(B, D, Action, Material, Support) :-
    slab_action(B, D, Action),
    slab_span(Action, Span, B, D),
    slab_material(Action, Span, Material),
    slab_support(Action, Material, Support).

slab_action(B, D, Action) :-
    Ratio is B / D,
    action_type(Action, X, Y),

```


Ratio > X,
Ratio =< Y.

slab_material(Action, Span, X) :-
 material_type(X, Y, Z),
 Span >= Y,
 Span =< Z,
 material_action(Action, X).

slab_support(Action, Material, Support) :-
 support_type(Material, Support),
 support_allowable(Action, Support).

slab_span(one_way, D, B, D).
slab_span(two_way, B, B, D).

Appendix C`

Frame Based Source Code for Slab Design

```

%*****
%
%               Initialize frames
%
%*****

(de start ()
  (init_grid_frame)
  (init_materials)
  (init_actions))

%*****

(df initialize (alstots)
  (let ((frame_name (second alstots))
        (frame_type (first alstots)))
    (fput frame_name 'ako 'value frame_type)
    (foreach slot in
      (fget-i frame_name 'template_list) do
        (prog ()
          (fmultput frame_name slot 'if-needed
            (fget frame_type slot 'if-needed))
          (fmultput frame_name slot 'if-added
            (fget frame_type slot 'if-added))
          (fmultput frame_name slot 'if-removed
            (fget frame_type slot 'if-removed))))))
  true)

%*****

(de init_grid_frame ()
  (fmultput 'grid_frame 'template_list 'value
    '(b d ratio material action_type
      edge_supports alternatives))
  (fmultput 'grid_frame 'b 'if-needed '(prompt_user))
  (fmultput 'grid_frame 'd 'if-needed '(prompt_user))
  (fmultput 'grid_frame 'ratio 'if-needed
    '(calculate_ratio))
  (fmultput 'grid_frame 'material 'if-needed
    '(select_material))

```

```

(fmultput 'grid_frame 'action_type 'if-needed
  '(select_action_type))
(fmultput 'grid_frame 'edge_supports
  'if-needed '(select_edge_supports))
(fmultput 'grid_frame 'alternatives 'if-needed
  '(select_alternatives))
true)

```

```

%*****

```

```

(de init_materials ()
  (fmultput 'materials
    'materials_list 'value '(steel_deck prefab_panels
      reinf_concrete
      prestr_concrete))
  (fput 'materials 'steel_deck 'action_type 'one-way)
  (fput 'materials 'steel_deck 'support 2)
  (fmultput 'materials
    'steel_deck 'span_range '(0.0 4.0))
  (fput 'materials 'prefab_panels 'action_type 'one-way)
  (fput 'materials 'prefab_panels 'support 2)
  (fmultput 'materials
    'prefab_panels 'span_range '(0.0 5.0))
  (fmultput 'materials
    'reinf_concrete 'action_type '(one-way two-way))
  (fmultput 'materials
    'reinf_concrete 'support '(0 2 4))
  (fmultput 'materials
    'reinf_concrete 'span_range '(3.0 8.0))
  (fmultput 'materials
    'prestr_concrete 'action_type '(one-way two-way))
  (fmultput 'materials
    'prestr_concrete 'support '(0 2 4))
  (fmultput 'materials
    'prestr_concrete 'span_range '(6.0 13.0))
  true)

```

```

%*****

```

```

(de init_actions ()
  (fmultput 'actions 'one-way 'support_cond '(0 2))
  (fmultput 'actions 'two-way 'support_cond '(0 2 4))
  (fput 'actions 'one-way 'span 'd)
  (fput 'actions 'two-way 'span 'b)
  true)

```

```

%*****
%
%               Define Procedural Attachments
%
%*****

(de select_alternatives ()
  (fget-v-d-f frame 'material)
  (foreach altern in (fget-v-d-f frame 'alternatives) do
    (let ((material (first altern))
          (action (second altern))
          (supp_list (third altern)))
      (myprint " ")
      (foreach supp in supp_list do
        (prin2 "Material: ")
        (myprint material)
        (prin2 "Action   ")
        (myprint action)
        (prin2 "Support  ")
        (myprint supp)
        (myprint " "))))

  true)

%*****

(de select_material ()
  (let ((material_list (fget 'materials 'materials_list
                             'value)))
    (foreach material in material_list do
      (foreach action in
        (fget-v-d-f frame 'action_type) do
        (if (member action
          (fget 'materials material 'action_type))
          (let ((span
            (first (fget frame
              (first (fget 'actions action 'span))
              'value))))
            (if (null (and
              (>= span
                (first
                  (fget 'materials material 'span_range)))
              (<= span
                (second
                  (fget 'materials material 'span_range))
                ))))
              false
              (fput frame 'material 'value material)
              (fput frame 'alternatives 'value

```

```

      (list material action
        (intersection
          (fget 'materials material 'support)
          (fget 'actions action 'support_cond)
          )))))))
    (fget-v-d-f frame 'material)))

%*****

(de select_action_type ()
  (let ((ratio (first (fget-v-d-f frame 'ratio))))
    (cond ((> ratio 1.5)
      (fput frame 'action_type 'value 'one-way))
      ((<= ratio 1.2)
      (fput frame 'action_type 'value 'two-way))
      (otherwise
      (fput frame 'action_type 'value 'one-way)
      (fput frame 'action_type 'value 'two-way)))
    (fget frame 'action_type 'value)))

%*****

(de select_edge_supports ()
  (foreach material in (fget-v-d-f frame 'material) do
    (foreach support_cond in
      (fget 'materials material 'support) do
      (fput frame 'edge_supports 'value support_cond)))
    (fget frame 'edge_supports 'value))

%*****

(de calculate_ratio ()
  (let ((b (first (fget-v-d-f frame 'b)))
        (d (first (fget-v-d-f frame 'd))))
    (fput frame 'ratio 'value (/ b d))
    (fget frame 'ratio 'value)))

%*****

(de prompt_user ()
  (writeln '(please enter a value for slot:) (list slot))
  (fput frame slot 'value (promptread "input>"))
  (fget frame slot 'value))

%*****

```

Appendix D

A Listing of Rules in MASON

This appendix provides a listing of all the distinct classes of rules incorporated into the MASON system. The syntax of the rules differs slightly from the syntax given in Chapter 4. The syntax presented in Chapter 4 represents the salient features of the production rules in EMASON. The following is a list of the actual rules used by the system and contain certain features that are intended for use at a low level of representation.

D.1 Special Rules

```
(RULE SR_1 (IF ((@# - (STATUS OF VARIABLE => LOCATION) 'UNBOUND) NIL))
  (THEN ((@# GET_PARAMETERS_FOR (GET_VAL LOCATION)) (SR1)))
  (MESSAGE_TEXT (RETRIEVING PARAMETERS FROM NATIONAL BUILDING CODE))
  (BOOLEAN_FN) (1.0))

(RULE SR_2 (IF ((@# - (STATUS OF VARIABLE => *FILES*) 'UNBOUND) NIL))
  (THEN ((@# BINDER '*FILES* TRUE) (SR_2))
    ((@# GET_MASONRY_PARAMETERS) (SR_2)) ((@# GET_CISC_INFO) (SR_2)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE SR_3 (IF ((@# - (STATUS OF VARIABLE => 'E_STEEL) 'UNBOUND) NIL))
  (THEN ((@# BINDER 'E_STEEL 200000) (SR_3))) (MESSAGE_TEXT NIL)
  (BOOLEAN_FN) (1.0))
```

D.2 General Rules

```
(RULE GR1
  (IF (($WALL IS CRACKED) NIL) (($WALL IS AN INFILL) NIL)
    (($WALL IS NON LOAD BEARING) NIL))
  (THEN ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) (GR1))
    ((SHRINKAGE OF $WALL SHOULD BE INVESTIGATED) (GR1)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE GR2
  (IF ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) (GR1 GR10))
    ((THE CRACKS APPEAR NEAR THE TOP OF THE WALL) NIL)
    ((THE UNITS ARE CRACKED) NIL))
  (THEN
```

```
(( $WALL HAS TAKEN ON SOME GRAVITY LOADS SHOULD BE INVESTIGATED) (GR2)))
(MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE GR3
```

```
(IF
  (( $WALL HAS TAKEN ON SOME GRAVITY LOADS SHOULD BE INVESTIGATED)
    (GR2 GR7))
  ((A SPACE HAS BEEN LEFT AT THE TOP OF $WALL) NIL))
(THEN ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3))) (MESSAGE_TEXT NIL)
(MINNER) (1.0))
```

```
(RULE GR4
```

```
(IF
  (( $WALL HAS TAKEN ON SOME GRAVITY LOADS SHOULD BE INVESTIGATED)
    (GR2 GR7))
  ((A SPACE HAS BEEN LEFT AT THE TOP OF $WALL) NIL)
  ((FLOOR IS BENDING PERPENDICULAR TO WALL) NIL))
(THEN ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR4))) (MESSAGE_TEXT NIL)
(STRENGTHEN) (1.0))
```

```
(RULE GR5
```

```
(IF
  (( $WALL HAS TAKEN ON SOME GRAVITY LOADS SHOULD BE INVESTIGATED)
    (GR2 GR7))
  ((A SPACE HAS BEEN LEFT AT THE TOP OF $WALL) NIL)
  ((FLOOR IS BENDING PARALLEL TO WALL) NIL))
(THEN ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5))) (MESSAGE_TEXT NIL)
(STRENGTHEN) (1.0))
```

```
(RULE GR6
```

```
(IF (( $WALL HAS TAKEN ON SOME GRAVITY LOADS SHOULD BE INVESTIGATED)
    (GR2 GR7))
  ((C$ <= (GET_VAL GAP_PRIME) 0.0) NIL)
  ((A SPACE HAS BEEN LEFT AT THE TOP OF $WALL) NIL)
  ((FLOOR IS BENDING PARALLEL TO WALL) NIL))
(THEN ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR6))) (MESSAGE_TEXT NIL)
(STRENGTHEN) (1.0))
```

```
(RULE GR7
```

```
(IF ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) (GR1))
  ((CRACKS APPEAR NEAR THE BOTTOM OF THE WALL) NIL)
  ((THE UNITS ARE CRACKED) NIL))
(THEN
  (( $WALL HAS TAKEN ON SOME GRAVITY LOADS SHOULD BE INVESTIGATED) (GR7)))
(MESSAGE_TEXT NIL) (STRENGTHEN) (1.0))
```

```
(RULE GR8
```

```
(IF ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) (GR1 GR10))
```

```
((CRACKS APPEAR AT THE CORNERS OF THE WALL) NIL))
(THEN ((WALL BEAM INTERACTION SHOULD BE INVESTIGATED) (GR8)))
(MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE GR9
  (IF ((WALL BEAM INTERACTION SHOULD BE INVESTIGATED) (GR8))
    ((THE UNITS ARE CRACKED) NIL))
  (THEN ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR9))) (MESSAGE_TEXT NIL)
  (STRENGTHEN) (1.0))
```

```
(RULE GR10
  (IF (($WALL IS CRACKED) NIL) (($WALL IS AN INFILL) NIL)
    ((~ $WALL IS NON LOAD BEARING) NIL))
  (THEN ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) (GR10)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE GR11
  (IF ((FRAME WALL INTERACTION SHOULD BE INVESTIGATED) (GR1 GR10))
    ((CRACKS APPEAR NEAR THE CENTER OF THE PANEL) NIL)
    ((CRACKS PASS THROUGH THE MORTAR) NIL))
  (THEN ((SHOULD INVESTIGATE LATERAL LOADING) (GR11)))
  (MESSAGE_TEXT
    (THE POSSIBILITY THAT LATERAL LOADING HAS CAUSED CRACKING IN THE PANEL
     SHOULD BE INVESTIGATED))
  (MINNER) (1.0))
```

```
(RULE GR12
  (IF ((SHOULD INVESTIGATE LATERAL LOADING) (GR11))
    ((THE CRACK ORIENTATIONS ARE DIAGONAL TO THE PANEL) NIL)
    ((~ THERE IS EVIDENCE OF CRUSHING AT THE CORNERS) NIL))
  (THEN ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12))) (MESSAGE_TEXT NIL)
  (MINNER) (0.75))
```

```
(RULE GR13
  (IF ((SHOULD INVESTIGATE LATERAL LOADING) (GR11))
    ((THE CRACK ORIENTATIONS ARE DIAGONAL TO THE PANEL) NIL)
    ((THERE IS EVIDENCE OF CRUSHING AT THE CORNERS) NIL))
  (THEN ((SHOULD INVESTIGATE FAILURE TYPE 4A) (GR13))) (MESSAGE_TEXT NIL)
  (MINNER) (1.0))
```

```
(RULE GR14
  (IF ((SHRINKAGE OF $WALL SHOULD BE INVESTIGATED) (GR16))
    ((CRACKS ARE ORIENTED IN THE VERTICAL DIRECTION) NIL)
    ((~ $BUILDING IS MORE THAN ONE YEAR OLD) NIL))
  (THEN ((POSSIBLE CAUSE OF CRACKING IS UNIT SHRINKAGE) (GR14)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE GR15
```



```

(IF ((POSSIBLE CAUSE OF CRACKING IS UNIT SHRINKAGE) (GR14))
  (($WALL IS A LONG WALL) (IDO31)) ((CRACKS PASS THROUGH THE MORTAR) NIL))
(THEN ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))) (MESSAGE_TEXT NIL)
(MINNER) (1.0))

```

```

(RULE GR16
  (IF (($WALL IS CRACKED) NIL) ((~ $WALL IS AN INFILL) NIL)
    ((~ $WALL IS NON LOAD BEARING) NIL))
  (THEN ((SHRINKAGE OF $WALL SHOULD BE INVESTIGATED) (GR16)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

D.3 Property Rules

```

(RULE PR1
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    (($~ = (STATUS OF VARIABLE => FC_PRIME) 'UNBOUND) NIL))
  (THEN
    (($~
      (PROG NIL
        (BLKLN)
        (MYPRINT "Could you supply a value for fc_prime")
        (MYPRINT "The ultimate strength of the concrete Y/N")
        (IF (EVAL (PROMPTREAD "--->"))
          (RETURN (FIFTH (BINDER 'FC_PRIME (PROMPTREAD "--->")))) (BLKLN)
          (MYPRINT "Alright I will assume one") (BLKLN)
          (RETURN (FIFTH (BINDER 'FC_PRIME 25))))))
      (PR1)))
    (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```

(RULE PR2
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    (($BUILDING IS MORE THAN ONE YEAR OLD) NIL))
  (THEN
    (($~ BINDER 'ESPIL_SHRINK_T (* 0.9 (GET_VAL ESPIL_SHRINK_ULT))) (PR2)))
  (MESSAGE_TEXT
    (IT IS ASSUMED THAT AFTER ONE YEAR 80 PERCENT OF THE SHRINKAGE HAS
      OCCURRED))
  (MINNER) (1.0))

```

```

(RULE PR3
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    (($BUILDING IS MORE THAN ONE YEAR OLD) NIL))
  (THEN (($~ BINDER 'CREEP_T (* 0.8 (GET_VAL CREEP_ULT))) (PR3)))
  (MESSAGE_TEXT
    (IT IS ASSUMED THAT AFTER ONE YEAR 90 PERCENT OF THE CREEP HAS OCCURRED))
  (MINNER) (1.0))

```

```

(RULE PR4 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
(THEN ((@B BINDER 'CREEP_ULT 2.35) (PR4)))
(MESSAGE_TEXT
  (THE ULTIMATE CREEP COEFFICIENT HAS BEEN ASSUMED TO BE !235**10-2))
(MINNER) (1.0))

(RULE PR5 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
(THEN ((@B BINDER 'ESPIL_SHRINK_ULT (/ 300 (** 10.0 6))) (PR5)))
(MESSAGE_TEXT
  (THE ULTIMATE SHRINKAGE COEFFICIENT HAS BEEN ASSUMED TO BE !300**10-6))
(MINNER) (1.0))

(RULE PR6
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    ((~ $BUILDING IS MORE THAN ONE YEAR OLD) NIL)
    ((CONCRETE HAS BEEN MOIST CURED) NIL))
  (THEN
    ((@B BINDER 'ESPIL_SHRINK_T
      (* (GET_VAL ESPIL_SHRINK_ULT)
        (/ (GET_VAL T_DAYS) (1+ (GET_VAL T_DAYS) 35))))
      (PR6)))
  (MESSAGE_TEXT
    (ESPIL_SHRINK_T HAS BEEN CALCULATED ACCORDING TO ACI COMMITTEE 209))
  (MINNER) (1.0))

(RULE PR7
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    ((~ $BUILDING IS MORE THAN ONE YEAR OLD) NIL)
    ((CONCRETE HAS BEEN STEAM CURED) NIL))
  (THEN
    ((@B BINDER 'ESPIL_SHRINK_T
      (* (GET_VAL ESPIL_SHRINK_ULT)
        (/ (GET_VAL T_DAYS) (1+ (GET_VAL T_DAYS) 55))))
      (PR7)))
  (MESSAGE_TEXT
    (ESPIL_SHRINK_T HAS BEEN CALCULATED ACCORDING TO ACI COMMITTEE 209))
  (MINNER) (1.0))

(RULE PR8
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    ((~ $BUILDING IS MORE THAN ONE YEAR OLD) NIL))
  (THEN
    ((@B BINDER 'CREEP_T
      (* (GET_VAL CREEP_ULT)
        (/ (** (GET_VAL T_DAYS) 0.5999999)
          (1+ (** (GET_VAL T_DAYS) 0.5999999) 10))))
      (PR8)))
  (MESSAGE_TEXT (CREEP_T HAS BEEN CALCULATED ACCORDING TO ACI COMMITTEE 209))

```

(MINNER) (1.0))

(RULE PR9

(IF ((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL)
 ((CONCRETE IS NORMAL WEIGHT CONCRETE) NIL))
 (THEN ((@B BINDER 'E_CON (* 5000.0 (SQRT (GET_VAL FC_PRIME)))) (PR9)))
 (MESSAGE_TEXT (TAKEN FROM CLAUSE !6-4-1 CSA !23-3-M77)) (MINNER) (1.0))

(RULE PR10

(IF ((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL)
 ((CONCRETE IS NORMAL WEIGHT CONCRETE) NIL))
 (THEN
 ((@B BINDER 'E_CON
 (* 0.0429998 (* (** (GET_VAL W_CON) 1.5) (SQRT (GET_VAL FC_PRIME))))))
 (PR10)))
 (MESSAGE_TEXT (TAKEN FROM CLAUSE !6-4-1 CSA !23-3-M77)) (MINNER) (1.0))

(RULE PR11 (IF ((CONCRETE IS NORMAL WEIGHT CONCRETE) NIL))

(THEN ((@B BINDER 'W_CON 2400) (PR11))) (MESSAGE_TEXT NIL) (BOOLEAN_FN)
 (1.0))

(RULE PR12 (IF ((CONCRETE IS NORMAL WEIGHT CONCRETE) NIL))

(THEN ((@B PROMPT_USER_2 "W_CON) (PR12))
 ((@B BINDER 'W_CON (SYSTEM_MENU_HANDLER 'W_CON)) (PR12)))
 (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE PR15

(IF ((@B = (GET_VAL \$UNITS) 'BRICK) NIL)
 ((@B = (GET_VAL \$MORTAR) 'M) NIL))
 (THEN
 ((@B BINDER 'FM_PRIME
 (GET_TABLE 'TABLE_4335#7 (LIST 'BRICK 'M (GET_VAL FB_PRIME))))
 (PR15)))
 (MESSAGE_TEXT (TAKEN FROM CSA S304-M78 !4-3-3-5)) (BOOLEAN_FN) (1.0))

(RULE PR16

(IF ((@B = (GET_VAL \$UNITS) 'BRICK) NIL)
 ((@B = (GET_VAL \$MORTAR) 'N) NIL))
 (THEN
 ((@B BINDER 'FM_PRIME
 (GET_TABLE 'TABLE_4335#7 (LIST 'BRICK 'N (GET_VAL FB_PRIME))))
 (PR16)))
 (MESSAGE_TEXT (TAKEN FROM CSA S304-M78 !4-3-3-5)) (BOOLEAN_FN) (1.0))

(RULE PR17

(IF ((@B = (GET_VAL \$UNITS) 'BRICK) NIL)
 ((@B = (GET_VAL \$MORTAR) 'S) NIL))
 (THEN

```

((@#~ BINDER 'FM_PRIME
  (GET_TABLE 'TABLE_4335#7 (LIST 'BRICK 'S (GET_VAL FB_PRIME))))
  (PR17)))
(MESSAGE_TEXT (TAKEN FROM CSA S304-M78 14-3-3-5)) (BOOLEAN_FN) (1.0))

(RULE PR18
  (IF ((@#~ = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((@#~ = (GET_VAL $MORTAR) 'N) NIL))
  (THEN
    ((@#~ BINDER 'FM_PRIME
      (GET_TABLE 'TABLE_4335#7 (LIST 'BLOCK 'N (GET_VAL FB_PRIME))))
      (PR18)))
    (MESSAGE_TEXT (TAKEN FROM CSA S304-M78 14-3-3-7)) (BOOLEAN_FN) (1.0))

(RULE PR19
  (IF ((@#~ = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((@#~ NULL (= (GET_VAL $MORTAR) 'N)) NIL))
  (THEN
    ((@#~ BINDER 'FM_PRIME
      (GET_TABLE 'TABLE_4335#7 (LIST 'BLOCK 'M_OR_S (GET_VAL FB_PRIME))))
      (PR19)))
    (MESSAGE_TEXT
      (MORTAR IS EITHER M OR S HOWEVER IT HAS NO EFFECT ON FM_PRIME TAKEN FROM
        CSA S304-M78 14-3-3-7))
    (BOOLEAN_FN) (1.0))

(RULE PR20 (IF ((@#~ = (STATUS OF VARIABLE => $UNITS) 'UNBOUND) NIL))
  (THEN
    ((@#~
      (PROG NIL
        (BLKLN)
        (MYPRINT "Are the units brick or block?")
        (RETURN (FIFTH (BINDER '$UNITS (PROMPTREAD "-->"))))))
      (PR20)))
    (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE PR21 (IF ((@#~ = (STATUS OF VARIABLE => $MORTAR) 'UNBOUND) NIL))
  (THEN
    ((@#~
      (PROG NIL
        (BLKLN)
        (MYPRINT "Is the mortar type N, M or S?")
        (RETURN (FIFTH (BINDER '$MORTAR (PROMPTREAD "-->"))))))
      (PR21)))
    (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE PR22 (IF ((@#~ = (STATUS OF VARIABLE => FB_PRIME) 'UNBOUND) NIL))
  (THEN

```

```

((00$`
  (PROG NIL
    (BLKLN)
    (MYPRINT
      "Could you provide an estimate of the compressive
      strength of the units? Y/N")
    (IF (EVAL (PROMPTREAD "-->"))
      (RETURN (FIFTH (BINDER 'FB_PRIME (PROMPTREAD "-->")))) (BLKLN)
      (MYPRINT "Alright I will assume one")
      (IF (= (GET_VAL $UNITS) 'BRICK)
        (RETURN (FIFTH (BINDER 'FB_PRIME 40.0)))
        (RETURN (FIFTH (BINDER 'FB_PRIME 20.0)))))))
  (PR22)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE PR23
  (IF ((00$` = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((00$` = (STATUS OF VARIABLE => *PROPS*) 'UNBOUND) NIL))
  (THEN
    ((00$`
      (PROG NIL
        (BLKLN)
        (MYPRINT "Could you supply the type of blocks used? Y/N")
        (MYPRINT "Choices are:")
        (BLKLN)
        (PRINT BLOCK_TYPE_LIST)
        (BLKLN)
        (IF (EVAL (PROMPTREAD "-->"))
          (GET_TABLE 'NOMIN_MET_SIZ_20 (LIST 'BLOCK (PROMPTREAD "-->")))
          (BLKLN) (MYPRINT "Alright I will assume Hollow Block")
          (GET_TABLE 'NOMIN_MET_SIZ_20 (LIST 'BLOCK 'HOLLOW)))
          (RETURN (FIFTH (BINDER '*PROPS* TRUE))))))
      (PR23)))
    (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE PR24 (IF ((00$` = (STATUS OF VARIABLE => E_WALL) 'UNBOUND) NIL))
  (THEN
    ((00$` BINDER 'E_WALL
      (FIRST (MINNER (LIST (* (GET_VAL FM_PRIME) 1000.0) 20000.0))))
      (PR24)))
    (MESSAGE_TEXT
      (E_WALL HAS BEEN CHOSEN AS THE LEAST OF 1000*FM_PRIME OR 20000 MPA))
    (MINNER) (1 0))

(RULE PR25 (IF ((00$` = (STATUS OF VARIABLE => E_WALL) 'UNBOUND) NIL))
  (THEN
    ((00$` BINDER 'G_WALL
      (FIRST (MINNER (LIST (* (GET_VAL FM_PRIME) 400.0) 8000.0))))

```

```

(PR25)))
(MESSAGE_TEXT
(G_WALL HAS BEEN CHOSEN AS THE LEAST OF '400*FM_PRIME OR 8000 MPA))
(MINNER) (1 0))

```

D.4 Masonry Rules

```

(RULE ID001
(IF ((A SPACE HAS BEEN LEFT AT THE TOP OF $WALL) NIL)
((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL))
(THEN
((($ BINDER 'GAP_PRIME
(!- (GET_VAL GAP)
(!+ (GET_VAL DELTA_ELAST_COL)
(!+ (GET_VAL DELTA_CREP_COL) (GET_VAL DELTA_SHRINK_COL))))))
(ID001)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID002 (IF (($WALL HAS BEEN GROUTED) NIL))
(THEN (($ BINDER 'WIDTH_WALL (GET_VAL WIDTH_UNIT)) (ID002)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID003 (IF ((~ $WALL HAS BEEN GROUTED) NIL))
(THEN
(($ BINDER 'WIDTH_WALL (* 2 (GET_VAL FACE_SHELL_THICKNESS))) (ID003)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID004
(IF ((~ $WALL HAS BEEN GROUTED) NIL) ((~ UNITS ARE LIGHT WEIGHT) NIL))
(THEN
(($ BINDER 'WT_WALL (* (GET_VAL WALL_MASS) (/ 9.8099998 1000.0 0)))
(ID004)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID005
(IF ((~ $WALL HAS BEEN GROUTED) NIL) ((UNITS ARE LIGHT WEIGHT) NIL))
(THEN
(($ BINDER 'WT_WALL (* (GET_VAL WALL_MASS_LT) (/ 9.8099998 1000 0)))
(ID005)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID006
(IF (($WALL HAS BEEN GROUTED) NIL) ((UNITS ARE LIGHT WEIGHT) NIL))
(THEN (($ BINDER 'WT_WALL (* 323 0 (/ 9.8099998 1000.0))) (ID006)))
(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

```

```

(RULE ID007
  (IF ((@ - (GET_VAL $UNITS) 'BLOCK) NIL)
    (@ - (GET_VAL $MORTAR) 'N) NIL))
  (THEN ((@ BINDER 'V_ALLOW_STR 0.16) (ID007)))
  (MESSAGE_TEXT
    (ALLOWABLE SHEAR IN CONCRETE BLOCK TAKEN CSAS304-M78 TABLE 5 P36))
  (MINNER) (1.0))

(RULE ID008
  (IF ((@ - (GET_VAL $UNITS) 'BLOCK) NIL)
    (@ NULL (= (GET_VAL $MORTAR) 'N)) NIL))
  (THEN ((@ BINDER 'V_ALLOW_STR 0.23) (ID008)))
  (MESSAGE_TEXT
    (ALLOWABLE SHEAR IN CONCRETE BLOCK TAKEN CSAS304-M78 TABLE 5 P36))
  (MINNER) (1.0))

(RULE ID009 (IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL))
  (THEN
    (@ BINDER 'V_LATERAL
      (* (GET_VAL ACCELER_RATIO)
        (* (GET_VAL SEISMIC_RESPONSE)
          (* (GET_VAL K_EARTHQUAKE)
            (* (GET_VAL IMPORTANCE_FACTOR)
              (* (GET_VAL FOUNDATION_FACTOR) (GET_VAL WEIGHT_FRAME)))))))
    (ID009)))
  (MESSAGE_TEXT (LATERAL LOAD CALCULATED FROM NBCC !4-1-9-1~4 PAGE148))
  (MINNER) (1.0))

(RULE ID010 (IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL))
  (THEN
    (@ BINDER 'SEISMIC_RESPONSE
      (IF (> (/ 0.5 (SQRT (GET_VAL T_PERIOD)))) 1.0) 1.0
      (/ 0.5 (SQRT (GET_VAL T_PERIOD))))
    (ID010)))
  (MESSAGE_TEXT (SEISMIC RESPONSE FACTOR CALCULATED FROM !4-1-9-1~5))
  (MINNER) (1.0))

(RULE ID011
  (IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
    ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL))
  (THEN (@ BINDER 'T_PERIOD (* 0.1 (GET_VAL NO_STORIES))) (ID011)))
  (MESSAGE_TEXT (T_PERIOD CALCULATED FROM !4-1-9-1~6)) (MINNER) (1.0))

(RULE ID012
  (IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
    ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL))
  (THEN
    (@ BINDER 'T_PERIOD

```

```
(/ (* 0.0899996 (GET_VAL H_FRAME)) (* 1000.0 (SQRT (GET_VAL D_1))))
(IDO12)))
```

```
(MESSAGE_TEXT (T_PERIOD CALCULATED FROM !4-1-9-1-6)) (MINNER) (1.0))
```

```
(RULE IDO13
```

```
(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
```

```
((($BUILDING IS A POST DISASTER BUILDING) NIL))
```

```
(THEN ((@ $ BINDER 'IMPORTANCE_FACTOR 1.3) (IDO13)))
```

```
(MESSAGE_TEXT (IMPORTANCE FACTOR OBTAINED FROM NBCC !4-1-9-1-9)) (MINNER)
(1.0))
```

```
(RULE IDO14
```

```
(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
```

```
((($BUILDING IS A POST DISASTER BUILDING) NIL))
```

```
(THEN ((@ $ BINDER 'IMPORTANCE_FACTOR 1.0) (IDO14)))
```

```
(MESSAGE_TEXT (IMPORTANCE FACTOR OBTAINED FROM NBCC !4-1-9-1-9)) (MINNER)
(1.0))
```

```
(RULE IDO15
```

```
(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
```

```
((STRUCTURE FOUNDATIONS ARE ON ROCK OR DENSE TO STIFF SOIL) NIL))
```

```
(THEN ((@ $ BINDER 'FOUNDATION_FACTOR 1.0) (IDO15)))
```

```
(MESSAGE_TEXT
```

```
(DETERMINE FOUNDATION FACTOR ACCORDING TO NBCC !4-1-9-1-10 TABLE
!4-1-9-B)).
```

```
(MINNER) (1.0))
```

```
(RULE IDO16
```

```
(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
```

```
((STRUCTURE FOUNDATIONS ARE ON COMPACT COARSE SOILS OR STIFF FINE
GRAINED SOILS)
NIL))
```

```
(THEN ((@ $ BINDER 'FOUNDATION_FACTOR 1.3) (IDO16)))
```

```
(MESSAGE_TEXT
```

```
(DETERMINE FOUNDATION FACTOR ACCORDING TO NBCC !4-1-9-1-10 TABLE
!4-1-9-B))
```

```
(MINNER) (1.0))
```

```
(RULE IDO17
```

```
(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
```

```
((STRUCTURE FOUNDATIONS ARE ON VERY LOOSE COARSE SOILS OR VERY SOFT FINE
GRAINED SOILS)
NIL))
```

```
(THEN ((@ $ BINDER 'FOUNDATION_FACTOR 1.5) (IDO17)))
```

```
(MESSAGE_TEXT
```

```
(DETERMINE FOUNDATION FACTOR ACCORDING TO NBCC !4-1-9-1-10 TABLE
!4-1-9-B))
```

```
(MINNER) (1.0))
```


(RULE ID018

(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING IS MOMENT RESISTING FRAME) NIL)
 ((FRAME HAS THE CAPACITY TO RESIST TOTAL LATERAL LOAD) NIL))
 (THEN ((@B\$ BINDER 'K_EARTHQUAKE 0.6999999) (ID018)))
 (MESSAGE_TEXT
 (DETERMINE K_EARTHQUAKE FROM NBCC '4-1-9-1-7 TABLE '4-1-9-A'))
 (MINNER) (1.0))

(RULE ID019

(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING IS MOMENT RESISTING FRAME) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING HAS FLEXURAL WALLS) NIL)
 ((SYSTEMS ARE DESIGNED TO INTERACT USING RELATIVE STIFFNESSES) NIL))
 (THEN ((@B\$ BINDER 'K_EARTHQUAKE 0.6999999) (ID019)))
 (MESSAGE_TEXT
 (DETERMINE K_EARTHQUAKE FROM NBCC '4-1-9-1-7 TABLE '4-1-9-A'))
 (MINNER) (1.0))

(RULE ID020

(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING IS MOMENT RESISTING FRAME) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING HAS FLEXURAL WALLS) NIL)
 ((SYSTEMS ARE DESIGNED TO INTERACT INDEPENDENTLY) NIL)
 ((THE FLEXURAL WALLS ARE MASONRY) NIL))
 (THEN ((@B\$ BINDER 'K_EARTHQUAKE 0.8) (ID020)))
 (MESSAGE_TEXT
 (DETERMINE K_EARTHQUAKE FROM NBCC '4-1-9-1-7 TABLE '4-1-9-A'))
 (MINNER) (1.0))

(RULE ID021

(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING IS MOMENT RESISTING FRAME) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING HAS FLEXURAL WALLS) NIL)
 ((SYSTEMS ARE DESIGNED TO INTERACT INDEPENDENTLY) NIL)
 ((THE FLEXURAL WALLS ARE MASONRY) NIL))
 (THEN ((@B\$ BINDER 'K_EARTHQUAKE 1.3) (ID021)))
 (MESSAGE_TEXT
 (DETERMINE K_EARTHQUAKE FROM NBCC '4-1-9-1-7 TABLE '4-1-9-A'))
 (MINNER) (1.0))

(RULE ID022

(IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL)
 ((SUPPORTING STRUCTURE OF \$BUILDING HAS NON STANDARD FRAME) NIL))
 (THEN ((@B\$ BINDER 'K_EARTHQUAKE 2.0) (ID022)))
 (MESSAGE_TEXT
 (DETERMINE K_EARTHQUAKE FROM NBCC '4-1-9-1-7 TABLE '4-1-9-A'))

(MINNER) (1 0))

(RULE ID023

(IF ((@# = (STATUS OF VARIABLE => \$AGGREGATE) 'UNBOUND) NIL)

((@# = (GET_VAL \$UNITS) 'BLOCK) NIL))

(THEN

((@#

(PROG NIL

(BLKLN)

(MYPRINT

"Can you provide the type of aggregate
that was used for the blocks? Y/N")

(IF (EVAL (PROMPTREAD "-->"))

(COND (T

(MYPRINT

"Choices are SG (sand gravel), EI (expanded shale).")

(MYPRINT " C (cinder) and ES (expanded slag) ")

(RETURN (FIFTH (BINDER '\$AGGREGATE (PROMPTREAD "-->"))))

(BLKLN)))

(MYPRINT "Alright I will assume one")

(RETURN (FIFTH (BINDER '\$AGGREGATE 'C)))))

(ID0023)))

(MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID024

(IF ((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL)

((COLUMN IS RECTANGULAR IN CROSS SECTION) NIL))

(THEN

((BINDER 'I_COLUMN (/ (* (GET_VAL B_COL) (** (GET_VAL H_COL) 3)) 12))

(ID024)))

(MESSAGE_TEXT (MOMENT OF INERTIA OF COLUMN IS $B \cdot H^3 / 12$)) (MINNER) (1 0))

(RULE ID025

(IF ((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL)

((COLUMN IS RECTANGULAR IN CROSS SECTION) NIL)

((BENDING IS STRONG AXIS BENDING) NIL))

(THEN ((BINDER 'H_COL (GET_VAL DEPTH_COLUMN)) (ID025))

((BINDER 'B_COL (GET_VAL WIDTH_COLUMN)) (ID025)))

(MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID026

(IF ((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL)

((COLUMN IS RECTANGULAR IN CROSS SECTION) NIL)

((BENDING IS WEAK AXIS BENDING) NIL))

(THEN ((BINDER 'H_COL (GET_VAL WIDTH_COLUMN)) (ID026))

((BINDER 'B_COL (GET_VAL DEPTH_COLUMN)) (ID026)))

(MESSAGE_TEXT NIL) (MINNER) (1 0))

```

(RULE ID027
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS STEEL) NIL)
    ((BENDING IS STRONG AXIS BENDING) NIL))
  (THEN
    ((@B BINDER 'I_COLUMN (GET_CISC_HDBK (GET_VAL DESIGNATION) 'IX))
      (ID027)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID028
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS STEEL) NIL)
    ((BENDING IS WEAK AXIS BENDING) NIL))
  (THEN
    ((@B BINDER 'I_COLUMN (GET_CISC_HDBK (GET_VAL DESIGNATION) 'IY))
      (ID028)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID028
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS STEEL) NIL)
    ((BENDING IS WEAK AXIS BENDING) NIL))
  (THEN
    ((@B BINDER 'I_COLUMN (GET_CISC_HDBK (GET_VAL DESIGNATION) 'IY))
      (ID028)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID029 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS STEEL) NIL))
  (THEN
    ((@B BINDER 'I_BEAM (GET_CISC_HDBK (GET_VAL DESIGNATION) 'IX)) (ID029)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID030 (IF ((THERE IS SOME EVIDENCE OF AN EARTHQUAKE) NIL))
  (THEN
    ((@B BINDER 'WEIGHT_FRAME
      (* (GET_VAL D_||)
      (* (GET_VAL D|-)
      (!+ (GET_VAL WEIGHT_KPA) (* 0 25 (* 0 8 (GET_VAL GRD_SNOW_LD))))))
      (ID030)))
  (MESSAGE_TEXT (WEIGHT OF FRAME PLUS 25 PERCENT OF DESIGN SNOW LOAD))
  (MINNER) (1 0))

(RULE ID031
  (IF ((@B <= (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 0 5) NIL))
  (THEN ((@WALL IS A LONG WALL) (ID031))) (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

D.5 Failure Type 1 Rules

```

(RULE ID101
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))

```

```
((LOCAL STRESS EXCEEDS ALLOWABLE WALL STRESS) (ID102)))
(THEN ((CONCLUDE FAILURE TYPE 1) (ID101)))
(MESSAGE_TEXT
  (LOCAL STRESSES HAVE EXCEEDED THE AVERAGE TENSILE SPLITTING STRESS OF
   THE WALL CAUSING LOCAL CRACKING))
(MINNER) (1 0))
```

```
(RULE ID102
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((@> (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
  (THEN ((LOCAL STRESS EXCEEDS ALLOWABLE WALL STRESS) (ID102)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID103
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((@ GET_VAL FORCE_WALL) NIL))
  (THEN
    ((@ BINDER 'SIGMA_WALL
      (* 3 (/ (GET_VAL FORCE_WALL) (* (GET_VAL AREA_WALL) 4))))
      (ID103)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID104
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((ROOF HAS TOUCHED TOP OF WALL DUE TO ELASTIC DEFORMATION) (ID106)))
  (THEN
    ((@ BINDER 'FORCE_WALL
      (* (!- (GET_VAL DELTA_ELAS_BEAM) (GET_VAL GAP_PRIME))
        (GET_VAL K_BEAM)))
      (ID104)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID106
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((ROOF HAS TOUCHED TOP OF WALL DUE TO CREEP DEFORMATION) (ID107)))
  (THEN
    ((@ BINDER 'FORCE_WALL
      (* 0.8 (* (GET_VAL DELTA_ELAS_BEAM) (GET_VAL K_BEAM))))
      (ID106)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID108
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((ROOF HAS TRANSFERRED LOAD TO WALL) (ID118 ID120))
    ((@> (GET_VAL DELTA_ELAS_BEAM) (GET_VAL GAP_PRIME)) NIL))
  (THEN ((ROOF HAS TOUCHED TOP OF WALL DUE TO ELASTIC DEFORMATION) (ID106)))
  (MESSAGE_TEXT
    (THE GAP LEFT AT THE TOP OF THE WALL WAS INSUFFICIENT TO ALLOW FOR THE
```

ELASTIC DEFLECTION OF THE ROOF))
(MINNER) (1 0))

(RULE ID107

(IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
((ROOF HAS TRANSFERRED LOAD TO WALL) (ID116 ID120))
((δ < (GET_VAL DELTA_ELAS_BEAM) (GET_VAL GAP_PRIME)) NIL))
(THEN ((ROOF HAS TOUCHED TOP OF WALL DUE TO CREEP DEFORMATION) (ID107)))
(MESSAGE_TEXT
(THE GAP LEFT AT THE TOP OF THE WALL WAS INSUFFICIENT TO ALLOW FOR THE
CREEP DEFLECTION OF THE ROOF))
(MINNER) (1 0))

(RULE ID108

(IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
((SUPPORTING STRUCTURE OF \$BUILDING IS MOMENT RESISTING FRAME) NIL)
((ROOF BEAM HAS A RECTANGULAR CROSS SECTION) NIL))
(THEN
((δ BINDER 'K_BEAM
(* 192
(* (GET_VAL E_BEAM) (/ (GET_VAL I_BEAM)
(** (GET_VAL LENGTH_BEAM) 3))))
(ID108)))
(MESSAGE_TXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID109 (IF ((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL))
(THEN

((δ
(PROG NIL
(MYPRINT
"Could you please enter an estimate of the column stresses")
(MYPRINT "Y/N")
(IF (EVAL (PROMPTREAD "-->"))
(RETURN (FIFTH (BINDER 'SIGMA_COLUMN (PROMPTREAD "-->"))))
(MYPRINT
"Alright I will assume that sigma_column is 45% of fc_prime")
(RETURN
(FIFTH (BINDER 'SIGMA_COLUMN (* 0 45 (GET_VAL FC_PRIME))))))
(ID109)))

(MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID110

(IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
((SUPPORTING STRUCTURE OF \$BUILDING IS CONCRETE) NIL))
(THEN
((δ BINDER 'DELTA_ELAST_COL
(* (GET_VAL SIGMA_COLUMN) (/ (GET_VAL LENGTH_COL) (GET_VAL E_CON))))
(ID110)))

```

(MESSAGE_TEXT
  (ESTIMATE OF THE ELASTIC SHORTENING IN THE COLUMN WAS OBTAINED BY
  DIVIDING THE COLUMN STRESS BY THE MODULUS OF ELASTICITY AND MULTIPLYING
  IT BY THE COLUMN LENGTH))
(MINNER) (1.0))

(RULE ID111
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
  (THEN
    ((@ BINDER 'DELTA_CREP_COL
      (* (GET_VAL DELTA_ELAST_COL) (GET_VAL CREEP_T)))
      (ID111)))
  (MESSAGE_TEXT
    (CREEP DEFLECTION OF COLUMNS HAS BEEN CALCULATE AS THE CREEP COEFFICIENT
    TIMES THE ELASTIC DEFLECTION))
  (MINNER) (1.0))

(RULE ID112
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
  (THEN
    ((@ BINDER 'DELTA_SHRINK_COL
      (* (GET_VAL ESPIL_SHRINK_T) (GET_VAL LENGTH_COL)))
      (ID112)))
  (MESSAGE_TEXT
    (SHRINKAGE DEFLECTION OF COLUMNS HAS BEEN CALCULATE AS THE SHRINKAGE
    COEFFICIENT TIMES THE LENGTH OF THE COLUMN))
  (MINNER) (1.0))

(RULE ID113 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
  (THEN ((@ BINDER 'E_BEAM (GET_VAL E_CON)) (ID113))) (MESSAGE_TEXT NIL)
  (BOOLEAN_FN) (1.0))

(RULE ID114 (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4)))
  (THEN
    ((@ BINDER 'DELTA_ELAS_BEAM
      (/ (* (GET_VAL W_LOAD_BEAM) (** (GET_VAL LENGTH_BEAM) 4))
      (* 384 (* (GET_VAL E_BEAM) (GET_VAL I_BEAM)))))
      (ID114)))
  (MESSAGE_TEXT
    (ELASTIC DEFLECTION CALCULATED AT THE CENTER OF THE BEAM USING SMALL
    DEFLECTION THEORY THE MODEL IS A FIXED-FIXED BEAM WITH A UDL))
  (BOOLEAN_FN) (1.0))

(RULE ID116
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))

```

```

(THEN
  ((@# BINDER 'DELTA_CREEP_BEAM
    (* (GET_VAL CREEP_T) (GET_VAL DELTA_ELAS_BEAM)))
    (ID115)))
(MESSAGE_TEXT
  (CREEP OF BEAM ESTIMATED AS CREEP_T*ELASTIC DEFLECTION OF BEAM))
(STRENGTHEN) (1.0))

```

```

(RULE ID116
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    ((@# > (GET_VAL GAP_PRIME) 0.0) NIL)
    ((@# < (GET_VAL GAP_PRIME)
      (!+ (GET_VAL DELTA_ELAS_BEAM) (GET_VAL DELTA_CREEP_BEAM)))
      NIL))
  (THEN ((ROOF HAS TRANSFERRED LOAD TO WALL) (ID116))) (MESSAGE_TEXT NIL)
  (BOOLEAN_FN) (1.0))

```

```

(RULE ID117
  (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
    ((BEAM HAS A RECTANGULAR CROSS SECTION) NIL))
  (THEN
    ((@# BINDER 'I_BEAM
      (/ (* (GET_VAL WIDTH_BEAM) (** (GET_VAL HEIGHT_BEAM) 3)) 12.0))
      (ID117)))
  (MESSAGE_TEXT (AN ESTIMATE OF THE STIFFNESS OF THE BEAM IS  $B \cdot H^{**3/12}$ ))
  (MINNER) (1.0))

```

```

(RULE ID118
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((FLOOR IS BENDING PERPENDICULAR TO WALL) NIL))
  (THEN ((@# BINDER 'D_WALL (GET_VAL WIDTH_BEAM)) (ID118)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```

(RULE ID119
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((FLOOR IS BENDING PARALLEL TO WALL) NIL))
  (THEN ((@# BINDER 'D_WALL (GET_VAL HEIGHT_BEAM)) (ID119)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```

(RULE ID120
  (IF ((@# <= (GET_VAL GAP_PRIME) 0.0) NIL)
    ((FLOOR IS BENDING PERPENDICULAR TO WALL) NIL))
  (THEN ((ROOF HAS TRANSFERRED LOAD TO WALL) (ID120))
    ((ROOF HAS TOUCHED TOP WALL DUE TO ELASTIC DEFORMATION) (ID120)))
  (MESSAGE_TEXT
    (CREEP DEFORMATION OF THE FRAME HAS CAUSED IT TO TOUCH THE TOP OF THE
    WALL))

```

```

(BOOLEAN_FN) (1.0))

(RULE ID121
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((@> (/ (GET_VAL LENGTH_WALL) (GET_VAL HEIGHT_WALL)) 1.5) NIL))
  (THEN ((@ BINDER 'SIGMA_ALLOW (* 0.5 (GET_VAL FM_PRIME))) (ID121)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE ID122 (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4)))
  (THEN
    ((@ BINDER 'AREA_WALL (* (GET_VAL D_WALL) (GET_VAL WIDTH_WALL)))
      (ID122)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

(RULE ID123
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 1) (GR3 GR4))
    ((@ <= (/ (GET_VAL LENGTH_WALL) (GET_VAL HEIGHT_WALL)) 1.5) NIL))
  (THEN ((@ BINDER 'SIGMA_ALLOW (* 0.75 (GET_VAL FM_PRIME))) (ID123)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1.0))

```

D.6 Failure Type 2 Rules

```

(RULE ID201
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5 GR6))
    ((OVERALL STRESS IN WALL EXCEEDS ALLOWABLE STRESS) (ID202)))
  (THEN ((CONCLUDE FAILURE TYPE 2) (ID201)))
  (MESSAGE_TEXT
    (OVERALL STRESSES IN THE WALL HAVE EXCEEDED THE COMPRESSIVE CAPACITY OF
     THE WALL PRODUCING TENSILE CRACKING IN THE WALL))
  (MINNER) (1.0))

(RULE ID202
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5 GR6))
    ((@ > (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
  (THEN ((OVERALL STRESS IN WALL EXCEEDS ALLOWABLE STRESS) (ID202)))
  (MESSAGE_TEXT NIL) (STRENGTHEN) (1.0))

(RULE ID203 (IF ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5 GR6)))
  (THEN
    ((@ BINDER 'SIGMA_WALL
      (/
        (* (/ (GET_VAL K_WALL) (!+ (GET_VAL K_WALL) (GET_VAL K_COLUMN)))
          (!+ (* (GET_VAL W_LOAD_WALL) (GET_VAL LENGTH_WALL))
            (* 2 (* (GET_VAL SIGMA_COLUMN) (GET_VAL AREA_COLUMN)))
            (GET_VAL AREA_COLUMN)))
        (GET_VAL AREA_WALL)))
      (ID203)))

```



```
(MESSAGE_TEXT
(WALL STRESS HAS BEEN CALCULATED AS THE STIFFNESS OF THE WALL TIMES THE
APPLIED LOAD DIVIDED BY THE COMBINED STIFFNESS OF THE WALL AND COLUMNS))
(MINNER) (1.0))
```

```
(RULE ID204 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
(THEN
((@BINDER 'AREA_TRANS_COLUMN
(* (GET_VAL AREA_COLUMN)
(1+ (!- 1 (GET_VAL RHO_COLUMN))
(* (GET_VAL E_STEEL) (/ (GET_VAL RHO_COLUMN) (GET_VAL E_CON))))))
(ID204)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID205
(IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL)
((COLUMN HAS A RECTANGULAR CROSS SECTION) NIL))
(THEN
((@BINDER 'AREA_COLUMN
(* (GET_VAL WIDTH_COLUMN) (GET_VAL DEPTH_COLUMN)))
(ID205)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID206 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
(THEN
((@BINDER
(PROG NIL
(BLKLN)
(MYPRINT
"Could estimate the percent of steel in the columns? Y/N")
(IF (EVAL (PROMPTREAD "-->"))
(RETURN (FIFTH (BINDER 'RHO_COLUMN (PROMPTREAD "-->")))) (BLKLN)
(MYPRINT "Ok I will assume 3% of gross area of columns") (BLKLN)
(RETURN (FIFTH (BINDER 'RHO_COLUMN 0 03)))))
(ID206)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID207 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
(THEN
((@BINDER 'K_COLUMN
(* (GET_VAL AREA_TRANS_COLUMN)
(/ (GET_VAL E_CON) (GET_VAL LENGTH_COL))))
(ID207)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID208 (IF ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5 GR6)))
(THEN
((@BINDER 'K_WALL
```

```

      (* (GET_VAL AREA_WALL) (/ (GET_VAL E_WALL) (GET_VAL HEIGHT_WALL))))
      (ID208)))
    (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID209
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5 GR6))
    ((FLOOR IS BENDING PARALLEL TO WALL) NIL))
  (THEN
    ((@# BINDER 'AREA_WALL (* (GET_VAL LENGTH_WALL) (GET_VAL WIDTH_WALL))
      (ID209)))
    (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID210
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 2) (GR5 GR6))
    ((CRACKS APPEAR NEAR THE BOTTOM OF THE WALL) NIL))
  (THEN
    ((@# BINDER 'W_LOAD_WALL (+ (GET_VAL WT_WALL) (GET_VAL W_LOAD_BEAM))
      (ID210)))
    (MESSAGE_TEXT NIL) (STRENGTHEN) (1 0))

(RULE ID211 (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR9))
  (THEN ((@# BINDER 'SIGMA_ALLOW (GET_VAL FM_PRIME)) (ID211)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

(RULE ID212
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR9))
    ((THE CRACKS APPEAR NEAR THE TOP OF THE WALL) NIL))
  (THEN ((@# W_LOAD_WALL (GET_VAL W_LOAD_BEAM)) (ID212)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

```

D.7 Failure Type 3 Rules

```

(RULE ID301
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR9))
    ((WALL STRESSES AT CORNERS EXCEED ALLOWABLE STRESS) (ID302)))
  (THEN ((CONCLUDE FAILURE TYPE 3) (ID301)))
  (MESSAGE_TEXT
    (WALL BEAM INTERACTION HAS CAUSED LARGE LOCAL STRESSES NEAR THE CORNERS
      OF THE WALL NEAR THE BEAM SUPPORTS THESE STRESSES EXCEED THE WALL
      COMPRESSIVE STRESSES))
  (STRENGTHEN) (1 0))

(RULE ID302
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR9))
    ((@# > (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
  (THEN ((WALL STRESSES AT CORNERS EXCEED ALLOWABLE STRESS) (ID302)))
  (MESSAGE_TEXT NIL) (BOOLEAN_FN) (1 0))

```

```

(RULE ID303 (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0)))
  (THEN
    ((C$~ BINDER 'SIGMA_MAX
      (/ (* (GET_VAL C1) (GET_VAL W_LOAD_WALL)) (GET_VAL EQ_WIDTH_UNIT)))
      (ID303)))
    (MESSAGE_TEXT (SIGMA_MAX CALCULATED AS C1 * W_LOAD_WALL / WIDTH_WALL))
    (BOOLEAN_FN) (1.0))

(RULE ID304 (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0)))
  (THEN ((C$~ BINDER 'C1 (!+ 1 (* (GET_VAL BETA) (GET_VAL R)))) (ID304)))
  (MESSAGE_TEXT (C1 CALCULATED AS 1 !+ BETA * R)) (MINNER) (1.0))

(RULE ID305
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
    ((C$~ > (GET_VAL R) 7) NIL))
  (THEN ((BEAM IS VERY FLEXIBLE) (ID305)))
  (MESSAGE_TEXT (TRIANGULAR STRESS DISTRIBUTION)) (MINNER) (1.0))

(RULE ID306
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
    ((C$~ <= (GET_VAL R) 7) NIL) ((C$~ > (GET_VAL R) 5) NIL))
  (THEN ((BEAM IS FLEXIBLE) (ID306)))
  (MESSAGE_TEXT (QUADRATIC STRESS DISTRIBUTION)) (MINNER) (1.0))

(RULE ID307
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
    ((C$~ <= (GET_VAL R) 5) NIL))
  (THEN ((BEAM IS STIFF) (ID307)))
  (MESSAGE_TEXT (CUBIC STRESS DISTRIBUTION)) (MINNER) (1.0))

(RULE ID308
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
    ((BEAM IS VERY FLEXIBLE) (ID306)))
  (THEN ((C$~ BINDER 'SIGMA_WALL (/ (GET_VAL SIGMA_MAX) 2 0)) (ID308)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID309
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
    ((BEAM IS FLEXIBLE) (ID306)))
  (THEN ((C$~ BINDER 'SIGMA_WALL (/ (GET_VAL SIGMA_MAX) 3 0)) (ID309)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID310
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0)) ((BEAM IS STIFF) (ID307)))
  (THEN ((C$~ BINDER 'SIGMA_WALL (/ (GET_VAL SIGMA_MAX) 4 0)) (ID310)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```
(RULE ID311 (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0)))
```

```
(THEN
```

```
((@B BINDER 'R
```

```
(SQRT
```

```
(SQRT
```

```
(/
```

```
(* (GET_VAL EQ_WIDTH_UNIT)
```

```
(* (GET_VAL E_WALL) (** (GET_VAL HEIGHT_WALL) 3)))
```

```
(* (GET_VAL I_BEAM) (GET_VAL E_BEAM))))))
```

```
(ID311)))
```

```
(MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE ID312
```

```
(IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
```

```
((@B >= (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 1.0) NIL))
```

```
(THEN ((@B BINDER 'BETA 1 1000008) (ID312))) (MESSAGE_TEXT NIL) (MINNER)
```

```
(1.0))
```

```
(RULE ID313
```

```
(IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
```

```
((@B >= (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 0.5) NIL)
```

```
((@B < (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 1.0) NIL))
```

```
(THEN
```

```
((@B BINDER 'BETA
```

```
(!- 2.0 (* 0.8 (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL))))))
```

```
(ID313)))
```

```
(MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE ID314
```

```
(IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
```

```
((@B >= (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 0.4) NIL)
```

```
((@B < (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 0.5) NIL))
```

```
(THEN
```

```
((@B BINDER 'BETA
```

```
(!- 3.6 (* 4.0 (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL))))))
```

```
(ID314)))
```

```
(MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE ID315
```

```
(IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0))
```

```
((@B < (/ (GET_VAL HEIGHT_WALL) (GET_VAL LENGTH_WALL)) 0.4) NIL))
```

```
(THEN ((@B BINDER 'BETA 2.0) (ID315))) (MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE ID316 (IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR0)))
```

```
(THEN ((@B BINDER 'SIGMA_ALLOW (* 0.2 (GET_VAL FM_PRIME))) (ID316)))
```

```
(MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```
(RULE ID317-
```

```

(IF ((SHOULD INVESTIGATE FAILURE TYPE 3) (GR9))
  ((~ $WALL IS NON LOAD BEARING) NIL))
(THEN
  ((@ $ BINDER 'W_LOAD_WALL
    (!+ (GET_VAL W_LOAD_BEAM)
      (* (/ (GET_VAL HEIGHT_WALL) 1000.0) (GET_VAL WT_WALL))))
    (ID317)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

D.8 Failure Type 4 Rules

```

(RULE ID401
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SHEAR IN $WALL EXCEEDS SHEAR CAPACITY OF WALL) (ID402)))
  (THEN ((CONCLUDE FAILURE TYPE 4) (ID401))) (MESSAGE_TEXT NIL) (MINNER)
  (1.0))

(RULE ID402
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((@ $ > (GET_VAL V_WALL) (GET_VAL V_ALLOW)) NIL))
  (THEN ((SHEAR IN $WALL EXCEEDS SHEAR CAPACITY OF WALL) (ID402)))
  (MESSAGE_TEXT
    (THE SHEAR CAPACITY OF THE WALL HAS BEEN EXCEEDED DUE TO LATERAL LOADING))
  (MINNER) (1.0))

```

```

(RULE ID403
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL)
    ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED AT THE BASE FRAME) NIL)
    ((@ $ = (GET_VAL NO_STORIES) 1) NIL))
  (THEN
    ((@ $ BINDER 'V_FRAME
      (/
        (* (GET_VAL E_FRAME)
          (* (GET_VAL I_COLUMN)
            (* (GET_VAL K_FRAME) (* (GET_VAL DELTA_FRAME) 12))))
        (* (** (GET_VAL H_FRAME) 3) (* (!+ (GET_VAL K_FRAME) 1) 2000.0))))
      (ID403)))
    (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```

(RULE ID404
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL)
    ((SUPPORTING STRUCTURE OF $BUILDING IS FIXED AT THE BASE FRAME) NIL)
    ((@ $ = (GET_VAL NO_STORIES) 1) NIL))
  (THEN
    ((@ $ BINDER 'V_FRAME

```

```

(/
  (* (GET_VAL E_FRAME)
    (* (GET_VAL I_COLUMN)
      (* 6 (* (+ (GET_VAL K_FRAME) 1) (GET_VAL DELTA_FRAME))))
      (* (** (GET_VAL H_FRAME) 3) (* 3 (+ (GET_VAL K_FRAME) 2000.0))))
    (ID404)))
(MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID405 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS STEEL) NIL))
  (THEN ((C$~ BINDER 'E_FRAME (GET_VAL E_STEEL)) (ID405)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID406 (IF ((SUPPORTING STRUCTURE OF $BUILDING IS CONCRETE) NIL))
  (THEN ((C$~ BINDER 'E_FRAME (GET_VAL E_CON)) (ID406))) (MESSAGE_TEXT NIL)
  (MINNER) (1.0))

(RULE ID407
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED FRAME) NIL)
    ((~ SUPPORTING STRUCTURE OF $BUILDING IS BRACED) NIL)
    ((~ FAILURE IS LOCALIZED TO END PANELS) NIL))
  (THEN
    ((C$~ BINDER 'V_WALL (/ (GET_VAL V_LATERAL) (GET_VAL NO_BAYS))) (ID407)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID408
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED FRAME) NIL)
    ((~ SUPPORTING STRUCTURE OF $BUILDING IS BRACED) NIL)
    ((FAILURE IS LOCALIZED TO END PANELS) NIL))
  (THEN ((C$~ BINDER 'V_WALL (/ (GET_VAL V_LATERAL) 2)) (ID408)))
  (MESSAGE_TEXT
    (FAILURE HAS BEEN LOCALIZED AND THE LATERAL LOAD HAS BEEN RESISTED BY
    THE END PANELS))
  (MINNER) (1.0))

(RULE ID409
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED FRAME) NIL)
    ((SUPPORTING STRUCTURE OF $BUILDING IS BRACED) NIL)
    ((~ FAILURE IS LOCALIZED TO END PANELS) NIL))
  (THEN
    ((C$~ BINDER 'V_WALL
      (/ (+ (GET_VAL V_LATERAL) (GET_VAL V_FRAME)) (GET_VAL NO_BAYS)))
      (ID409)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID410

```

```

(IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
  ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED FRAME) NIL)
  ((SUPPORTING STRUCTURE OF $BUILDING IS BRACED) NIL)
  ((FAILURE IS LOCALIZED TO END PANELS) NIL))
(THEN
  ((cs BINDER 'V_WALL
    (!- (/ (GET_VAL V_LATERAL) (* 2 (GET_VAL NO_BAYS))) (GET_VAL V_FRAME)))
    (ID410)))
  (MESSAGE_TEXT
    (FAILURE HAS BEEN LOCALIZED AND THE LATERAL LOAD HAS BEEN RESISTED BY
    THE END PANELS))
  (MINNER) (1.0))

```

```

(RULE ID411
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL)
    ((FAILURE IS LOCALIZED TO END PANELS) NIL))
  (THEN
    ((cs BINDER 'V_WALL
      (!- (/ (GET_VAL V_LATERAL) (GET_VAL NO_BAYS)) (GET_VAL V_FRAME)))
      (ID411)))
    (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```

(RULE ID412
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL)
    ((FAILURE IS LOCALIZED TO END PANELS) NIL))
  (THEN
    ((cs BINDER 'V_WALL (/ (!- (GET_VAL V_LATERAL) (GET_VAL V_FRAME)) 2))
      (ID412)))
    (MESSAGE_TEXT
      (FAILURE HAS BEEN LOCALIZED AND THE LATERAL LOAD HAS BEEN RESISTED BY
      THE END PANELS))
    (MINNER) (1.0))

```

```

(RULE ID413
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((cs > (GET_VAL NO_BAYS) 1) NIL))
  (THEN ((cs BINDER 'I_C (/ (GET_VAL I_COLUMN) 2)) (ID413)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

```

```

(RULE ID414
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
    ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED FRAME) NIL)
    ((SUPPORTING STRUCTURE OF $BUILDING IS BRACED) NIL))
  (THEN
    ((cs BINDER 'V_FRAME
      (/ (* (GET_VAL K_FRAME) (GET_VAL DELTA_FRAME)) 1000.0))

```

```

(ID414)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID415
(IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
  ((= (GET_VAL NO_STORIES) 1) NIL)
  ((SUPPORTING STRUCTURE OF $BUILDING IS MOMENT RESISTING FRAME) NIL))
(THEN
  ((CS BINDER 'K_FRAME
    (/ (* (GET_VAL I_BEAM) (GET_VAL H_FRAME))
      (* (GET_VAL I_C) (GET_VAL L_FRAME))))
  (ID415)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID416
(IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
  ((SUPPORTING STRUCTURE OF $BUILDING IS PINNED FRAME) NIL)
  ((SUPPORTING STRUCTURE OF $BUILDING IS BRACED) NIL))
(THEN
  ((CS BINDER 'K_FRAME
    (/
      (* (GET_VAL AREA_BRACE)
        (* (GET_VAL E_FRAME) (** (GET_VAL L_FRAME) 2)))
      (** (!+ (** (GET_VAL L_FRAME) 2) (** (GET_VAL H_FRAME) 2)) 1 5)))
  (ID416)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID417
(IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
  ((CS = (GET_VAL NO_STORIES) 1) NIL))
(THEN
  ((CS BINDER 'DELTA_FRAME
    (/ (* 1.1999998 (* (* (GET_VAL V_ALLOW) 1000 0) (GET_VAL H_FRAME))
      (* (GET_VAL L_FRAME) (* (GET_VAL EQ_WIDTH_UNIT) (GET_VAL G_WALL))))
  (ID417)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID418
(IF ((SHOULD INVESTIGATE FAILURE TYPE 4) (GR12 ID420))
  ((CS = (GET_VAL NO_STORIES) 1) NIL))
(THEN
  ((CS BINDER 'V_ALLOW
    (* (GET_VAL L_FRAME)
      (/ (* (GET_VAL EQ_WIDTH_UNIT) (GET_VAL V_ALLOW_STR)) 1000.0)))
  (ID418)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID419 (IF ((SHOULD INVESTIGATE FAILURE TYPE 14A) (GR13)))

```



```

(THEN
  ((@ BINDER 'F_BEARING_1
    (/ (* 1000 0 (GET_VAL V_PANEL))
      (* (GET_VAL ALPHA) (* (GET_VAL EQ_WIDTH_UNIT) (GET_VAL H_FRAME))))))
  (ID419))
  ((@ BINDER 'F_BEARING_2
    (/ (* 3 (* (GET_VAL H_FRAME) (* 1000 0 (GET_VAL V_PANEL))))
      (* (GET_VAL EQ_WIDTH_UNIT) (** (GET_VAL L_FRAME) 2))))
  (ID419))
  ((@ BINDER 'F_BEARING_MAX
    (FIRST (MAXER (LIST (GET_VAL F_BEARING_1) (GET_VAL F_BEARING_2))))))
  (ID419)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID420
  (IF ((SHOULD INVESTIGATE FAILURE TYPE '4A) (GR13))
    ((@ > (GET_VAL F_BEARING_MAX) (GET_VAL SIGMA_ALLOW)) NIL))
  (THEN ((SHOULD INVESTIGATE FAILURE TYPE 4) (ID420)))
  (MESSAGE_TEXT
    (LOCAL CRUSHING HAS BEEN CAUSED BY EXCEEDING THE ALLOWABLE STRESS AT THE
      CORNERS OF THE WALL))
  (MINNER) (1 0))

```

```

(RULE ID421
  (IF ((SHOULD INVESTIGATE FAILURE TYPE '4A) (GR13))
    ((@ <= (GET_VAL F_BEARING_MAX) (GET_VAL SIGMA_ALLOW)) NIL))
  (THEN
    ((LOCAL CRUSHING HAS BEEN CAUSED BY HAMMERING OF THE FRAME ON THE WALL)
      (ID421))
    ((INVESTIGATE FAILURE TYPE 4) (ID421)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID422 (IF ((SHOULD INVESTIGATE FAILURE TYPE '4A) (GR13)))
  (THEN ((@ BINDER 'SIGMA_ALLOW (* 0 2 (GET_VAL FM_PRIME))) (ID422)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID423 (IF ((SHOULD INVESTIGATE FAILURE TYPE '4A) (GR13)))
  (THEN
    ((@ BINDER 'ALPHA
      (/ (GET_VAL L_FRAME)
        (* 4 (SQRT ((+ (** (GET_VAL L_FRAME) 2) (** (GET_VAL H_FRAME) 2))))))
      (ID423)))
    (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID424
  (IF ((SHOULD INVESTIGATE FAILURE TYPE '4A) (GR13))
    ((FAILURE IS LOCALIZED TO END PANELS) NIL))
  (THEN

```

```
((08" BINDER 'V_PANEL (/ (GET_VAL V_LATERAL) (* 2 0 (GET_VAL NO_BAYS))))
(ID424)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID425
(IF ((SHOULD INVESTIGATE FAILURE TYPE '4A) (GR13))
  ((~ FAILURE IS LOCALIZED TO END PANELS) NIL))
(THEN
  ((08" BINDER 'V_PANEL (/ (GET_VAL V_LATERAL) (GET_VAL NO_BAYS)))
  (ID425)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

D.9 Failure Type 5 Rules

```
(RULE ID501
(IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
  ((TENSION STRESSES IN WALL EXCEED ALLOWABLE STRESS)
  (ID502 ID503 ID504 ID505)))
(THEN ((CONCLUDE FAILURE TYPE 5) (ID501))) (MESSAGE_TEXT NIL) (MINNER)
(1 0))
```

```
(RULE ID502
(IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
  ((CRACKS APPEAR AT THE COLUMN) NIL)
  (($WALL IS CONNECTED TO THE COLUMN) NIL)
  ((08" > (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
(THEN ((TENSION STRESSES IN WALL EXCEED ALLOWABLE STRESS) (ID502))
  ((CONNECTION TO THE COLUMN WAS IMPROPERLY DETAILED) (ID502)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID503
(IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
  ((CRACKS APPEAR NEAR THE MIDDLE OF $WALL) NIL)
  (($WALL IS RESTRAINED AT THE ENDS) NIL)
  ((~ THERE ARE MOVEMENT OR EXPANSION JOINTS IN $WALL) NIL)
  ((08" > (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
(THEN ((TENSION STRESSES IN WALL EXCEED ALLOWABLE STRESS) (ID503))
  ((EXPANSION OR MOVEMENT JOINTS SHOULD HAVE BEEN INCLUDED IN $WALL)
  (ID503)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID504
(IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
  ((CRACKS APPEAR NEAR THE MIDDLE OF $WALL) NIL)
  ((~ $WALL HAS OPENINGS) NIL)
  ((THERE ARE MOVEMENT OR EXPANSION JOINTS IN $WALL) NIL)
  ((08" > (GET_VAL SPACING) 6) NIL)
```

```
((00" > (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
(THEN ((TENSION STRESSES IN WALL EXCEED ALLOWABLE STRESS) (ID504))
  ((EXPANSION OR MOVEMENT JOINTS WERE SPACED TOO FAR APART) (ID504)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID505
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    (($WALL HAS OPENINGS) NIL)
    ((CRACKS APPEAR NEAR THE OPENINGS OF $WALL) NIL)
    ((00" > (GET_VAL SIGMA_WALL) (GET_VAL SIGMA_ALLOW)) NIL))
  (THEN ((TENSION STRESSES IN WALL EXCEED ALLOWABLE STRESS) (ID505))
    ((OPENINGS WERE IMPROPERLY DETAILED) (ID505)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))
```

```
(RULE ID506 (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15)))
  (THEN
    ((00" BINDER 'SIGMA_WALL
      (/ (* (GET_VAL DELTA_WALL) (GET_VAL E_WALL))
        (+ (GET_VAL L_WALL)
          (/ (* (* 2 (GET_VAL AREA_WALL)) (GET_VAL E_WALL))
            (GET_VAL K_CONNECT))))))
    (ID506)))
  (MESSAGE_TEXT
    (STRESS IN WALL EQUALS LENGTH TIMES SHRINKAGE DIVIDED BY AREA TIMES
      MODULUS OF ELASTICITY))
  (MINNER) (1 0))
```

```
(RULE ID507 (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15)))
  (THEN
    ((00" BINDER 'DELTA_WALL
      (* (/ (GET_VAL L_WALL) 100.0) (GET_VAL EPSILON_SH_WALL)))
    (ID507)))
  (MESSAGE_TEXT
    (DEFLECTION OF WALL CALCULATED AS LENGTH TIMES SHRINKAGE VALUE FOR
      ASSEMBLAGE))
  (MINNER) (1.0))
```

```
(RULE ID508
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((00" = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((BLOCK HAS BEEN HIGH PRESSURE STEAM CURED) NIL)
    ((BLOCK WAS SATURATED WHEN LAID) NIL))
  (THEN
    ((00" BINDER 'EPSILON_SH_WALL
      (GET_TABLE 'TABLE_DS-3-21 (LIST 'HP 'SAT (GET_VAL $AGGREGATE))))
    (ID508)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))
```

```

(RULE ID509
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((@ = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((' BLOCK HAS BEEN HIGH PRESSURE STEAM CURED) NIL)
    ((BLOCK WAS SATURATED WHEN LAID) NIL))
  (THEN
    ((@ BINDER 'EPSILON_SH_WALL
      (GET_TABLE 'TABLE_DS-3-21 (LIST 'HT 'SAT (GET_VAL $AGGREGATE))))
      (ID509)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID510
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((@ = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((BLOCK HAS BEEN HIGH PRESSURE STEAM CURED) NIL)
    ((' BLOCK WAS SATURATED WHEN LAID) NIL))
  (THEN
    ((@ BINDER 'EPSILON_SH_WALL
      (GET_TABLE 'TABLE_DS-3-21 (LIST 'HP 'RH70 (GET_VAL $AGGREGATE))))
      (ID510)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID511
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((@ = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((' BLOCK HAS BEEN HIGH PRESSURE STEAM CURED) NIL)
    ((' BLOCK WAS SATURATED WHEN LAID) NIL))
  (THEN
    ((@ BINDER 'EPSILON_SH_WALL
      (GET_TABLE 'TABLE_DS-3-21 (LIST 'HT 'RH70 (GET_VAL $AGGREGATE))))
      (ID511)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

```

```

(RULE ID512
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((@ = (GET_VAL $UNITS) 'BLOCK) NIL) ((BOND PATTERN IS STACK BOND) NIL)
    ((' $WALL HAS BEEN GROUTED) NIL) ((@ = (GET_VAL $MORTAR) 'N) NIL))
  (THEN ((@ BINDER 'SIGMA_ALLOW 0.11) (ID512)))
  (MESSAGE_TEXT (TAKEN FROM TABLE 7 P36 CSA304)) (MINNER) (1.0))

```

```

(RULE ID513
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((@ = (GET_VAL $UNITS) 'BLOCK) NIL) ((BOND PATTERN IS STACK BOND) NIL)
    ((' $WALL HAS BEEN GROUTED) NIL)
    ((@ OR (= (GET_VAL $MORTAR) 'M) (= (GET_VAL $MORTAR) 'S)) NIL))
  (THEN ((@ BINDER 'SIGMA_ALLOW 0.16) (ID513)))
  (MESSAGE_TEXT (TAKEN FROM TABLE 7 P36 CSA304)) (MINNER) (1.0))

```

```

(RULE ID514
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR45))
    ((@# = (GET_VAL $UNITS) 'BLOCK) NIL)
    ((BOND PATTERN IS RUNNING BOND) NIL) ((~ $WALL HAS BEEN GROUTED) NIL)
    ((@# OR (= (GET_VAL $MORTAR) 'M) (= (GET_VAL $MORTAR) 'S)) NIL))
  (THEN ((@# BINDER 'SIGMA_ALLOW 0 32) (ID514)))
  (MESSAGE_TEXT (TAKEN FROM TABLE 7 P36 CSA304)) (MINNER) (1 0))

(RULE ID515
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((@# = (GET_VAL $UNITS) 'BLOCK) NIL) ((~ $WALL HAS BEEN GROUTED) NIL)
    ((BOND PATTERN IS RUNNING BOND) NIL) ((@# = (GET_VAL $MORTAR) 'N) NIL))
  (THEN ((@# BINDER 'SIGMA_ALLOW 0 22) (ID515)))
  (MESSAGE_TEXT (TAKEN FROM TABLE 7 P36 CSA304)) (MINNER) (1 0))

(RULE ID516
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 6) (GR15))
    ((~ THERE ARE MOVEMENT OR EXPANSION JOINTS IN $WALL) NIL))
  (THEN ((@# BINDER 'L_WALL (GET_VAL LENGTH_WALL)) (ID516)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID517
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((THERE ARE MOVEMENT OR EXPANSION JOINTS IN $WALL) NIL)
    ((@# > (GET_VAL SPACING) 6 0) NIL))
  (THEN ((@# BINDER 'L_WALL (GET_VAL SPACING)) (ID517)))
  (MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID518
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    (($WALL HAS OPENINGS) NIL)
    ((CRACKS APPEAR NEAR THE OPENINGS OF $WALL) NIL))
  (THEN
    ((@# BINDER 'H_WALL (!- (GET_VAL HEIGHT_WALL) (GET_VAL HEIGHT_OPENING)))
      (ID518)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID519
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    ((~ $WALL HAS OPENINGS) NIL))
  (THEN ((@# BINDER 'H_WALL (GET_VAL HEIGHT_WALL)) (ID519)))
  (MESSAGE_TEXT NIL) (MINNER) (1.0))

(RULE ID520
  (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15))
    (($WALL HAS OPENINGS) NIL)
    ((~ CRACKS APPEAR NEAR THE OPENINGS OF $WALL) NIL))
  (THEN ((@# BINDER 'H_WALL (GET_VAL HEIGHT_WALL)) (ID520)))

```

(MESSAGE_TEXT NIL) (MINNER) (1 0))

(RULE ID521 (IF ((SHOULD INVESTIGATE FAILURE TYPE 5) (GR15)))
(THEN
 ((@B BINDER AREA_WALL
 (* (* (GET_VAL H_WALL) 2) (GET_VAL FACE_SHELL_THICKNESS)))
 (ID521)))
(MESSAGE_TEXT NIL) (MINNER) (1 0))

Glossary

A more detailed glossary of the terminology used in the field of expert systems is provided by Hayes-Roth [29]. A substantial portion of this glossary was derived from that source.

access oriented. In the access oriented style of programming actions are potentially triggered when the data is accessed. Procedures are attached to data structures and are executed if certain predetermined conditions are satisfied. Frames are an example of access oriented programming.

assertion. A statement or claim that has yet to be substantiated.

backtracking. A search procedure that makes guesses at various points. If an unacceptable conclusion is reached the system returns to the point at which the guess was made and makes another guess.

backward chaining. A problem solving strategy that attempts to satisfy a goal recursively, first by determining the antecedent conditions necessary for its satisfaction, then by establishing the antecedent conditions as goals.

blackboard. A globally accessible data base used for recording immediate, partial results of problem solving strategies. The blackboard coordinates the activities of various sub-expert or specialist systems.

certainty factor. A measure of confidence, placed by the user or the expert system, in a hypothesis, assertion, or inferential rule.

context. A dynamic data base that contains a representation of the current state of the problem at hand. A state may be a set of variables and their values, some assertions waiting to be proved, or a collection of known facts.

control strategies. See problem solving strategies.

deep knowledge. Knowledge that is complete and has a theoretical or well understood basis, such as the knowledge of elasticity or structural behavior. This is opposed to heuristic knowledge.

expert system. A computer program that achieves high level of performance in a domain that for human beings would require years of training. This term is generally not applied to applications type programs that rely predominately on numerical evaluation, such as finite element programs. (Also knowledge based expert system.)

explanation. Motivating, justifying, or rationalizing a concept, hypothesis, heuristic rule, or law.

fire. An inferential rule is said to fire when all of the antecedent conditions are satisfied and at least one consequent action is left unperformed.

forward chaining. A problem solving strategy that recursively deduces consequent propositions from antecedent conditions. As new propositions are deduced new inferential rules are applied recursively.

frame. A knowledge representation scheme that associates the attributes of an object or of a class of objects in terms of various slots and slot-values. Slots may correspond to intrinsic features such as name, or definition, or may represent derived features such as values.

framework. Programming systems that simplify the task of building expert systems and the knowledge acquisition phase.

fuzzy sets. A knowledge representation technique that permits gradations of truth or membership. The expression of ambiguous facts and inferences is achieved through a membership function.

heuristic. See heuristic knowledge.

heuristic knowledge. Knowledge in the form of procedural tips (rules of thumb), or incomplete methods for performing some task.

inexact reasoning. A method of reasoning that takes into account uncertain or unreliable information, as opposed to formal reasoning methods that require boolean type responses.

inference mechanism. See problem solving strategies. Also referred to as inference engine.

inferential rule. An association between consequent beliefs and antecedent conditions that allows the consequents to be deduced provided the antecedent conditions hold.

knowledge acquisition. The extraction and formulation of knowledge from sources such as codes, specifications, references and especially experts.

knowledge base. The repository of knowledge that contains domain specific knowledge in a computer system.

knowledge paradigm. A method of programming that relies on domain knowledge focused searches, heuristics and symbolic evaluation. This method is well suited to ill structured domains.

meta-rule. A rule that determines the manner in which other rules should be applied.

object oriented. A method of programming that is based on an asynchronous message passing process. State changes occur when a message is received by an object. Objects have their own private procedures and data structures and are able to inherit states from a parent class or generic objects.

power paradigm. A method of programming that uses general inference procedures that rely on numerical evaluation. This method is useful for solving problems in a well structured domain—the canonical example being structural analysis.

problem solving strategy. A heuristic for control, that may or may not exploit domain specific knowledge. The heuristic or strategy is used to avoid exhaustive and possible combinatorial searches. Backwards chaining is an example of a problem solving strategy.

procedural attachment. The association of a procedure with a declarative data structure to which it applies. This pertains particularly to frames.

procedure oriented. A method of programming where operations and functions are defined as procedures. Procedures are invoked whenever they are called regardless of the data or information passed to them.

production rule. See **inferential rule**

reason maintenance. The task of preserving consistent beliefs in a reasoning system whose beliefs change over time. (Also **truth maintenance**)

schema. See **frame**

self reasoning. See **explanation**

semantic nets. A knowledge representation scheme that relates the meaning, intention or significance of symbolic expressions rather than relating their form, structure, or attributes.

state space. A method of representation that views a task as a series of states spaces. A state space is a list of acceptable or legal configurations of the components involved in the task. An example of a state in a state space problem would be the chess problem in a newspaper.

well formed formula. A collection of clauses, each of which is a collection of literals. The literals are combined using connectives such as $\wedge$ (and), $\vee$ (or), and $\rightarrow$ (implication). A literal is either an atomic formula or a negated atomic formula.

References

The following abbreviations are used throughout this list of references

- IJCAI77 *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977
 - IJCAI79 *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, 1979
 - IJCAI81 *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada, 1981
- 1 ACI Committee 209, *Predictions of Creep, Shrinkage and Temperature Effects in Concrete Structures*, ACI Publication SP 27, American Concrete Institute, Detroit, 1971
 - 2 Ang, A H-S and Tang, W H, *Probability Concepts in Engineering Planning and Design Volume I- Basic Principles*, John Wiley and Sons, 1975.
 - 3 Barnett, J A., "Computational Methods for a Mathematical Theory of Evidence", IJCAI81 pp 868-875, 1981
 - 4 Bennett, J S and Engelmores, R S, "SACON: A Knowledge-based Consultant for Structural Analysis", IJCAI79, pp 47-49, 1979
 - 5 Bennett, J S and Hollander, C.R., "DART: An Expert System for Computer Fault Diagnoses", IJCAI81, pp. 843-845, 1981
 - 6 Bobrow, D and Stefik, M., *The LOOPS Manual*, Xerox Corporation, 3333 Coyote Hill Road, Palo Alto, CA 94304, 1983.
 - 7 Bramer, M.A., "A Survey and Critical Review of Expert Systems Research", *Studies in Cybernetics: 1, Introductory Readings in Expert Systems*, D. Michie (ed.), pp. 3-29, Gordon and Breach, New York, 1982.
 - 8 Buchanan, B.G. and Feigenbaum, E.A., "DENDRAL and Meta-DENDRAL: Their Applications Dimension", *Artificial Intelligence*, 11, 1-2, pp. 5-24, 1978
 - 9 Bulger, S., *Cracking and Movement Joint Requirements in Low Rise Masonry Structures*, Fourth Year Project Report, Department of Civil Engineering, Carleton University, Ottawa, 1982.
 - 10 Byrne, D. and Eno, B., "Crosseyed and Painless," *Remain in Light*, Sire Records, XSR-6095, Index Music/Bleu Music Co. Inc., 1980.
 - 11 Canadian Standards Association, *Code for the Design of Concrete Structures for Buildings*, CSA Standard CAN3-A23.3-M77, 1977.
 - 12 Canadian Standards Association, *Masonry Construction for Buildings*, CSA Standard A371-M1980, 1980.
 - 13 Canadian Standards Association, *Masonry Design and Construction for Buildings*, CSA Standard CAN3-S304-M78, 1978.
 - 14 Canadian Standards Association, *Steel Structures for Buildings — Limit States Design*, CSA Standard CAN3-S16.1-M78, 1978.

- 15 Carter, C and Stafford-Smith, B, "Structural Behavior of Masonry Infilled Walls Subjected to Racking Loads", *Designing, Engineering and Construction with Masonry Products* Johnson, E.B. (ed), Gulf Publishing Company, pp 226-233, 1969
- 16 Clocksin, W F. and Mellish, C S, *Programming in PROLOG*, Springer Verlag 1981
- 17 Davies, S.R., and Ahmed, A.F., "An Approximate Method for Analyzing Composite Wall-Beams", *Proceedings of the British Ceramic Society*, No 27, December, 1978
- 18 Davis, R., Buchanan, B.G and Shortliffe, E.H., "Production Systems as a Representation for a Knowledge-based Consultation Program", *Artificial Intelligence*, 8, 1, pp 15-45, 1977
- 19 Davis, R., "Interactive Transfer of Expertise I Acquisition of New Inference Rules", *IJCAI77*, pp 321-328 1977
- 20 Doyle, J., *Truth Maintenance Systems for Problem Solving*, Technical Report AI-TR-419, MIT AI Laboratory, Cambridge, Massachusetts, 1978
- 21 Drysdale, R.G., and Suter, G.T., *Modern Engineered Masonry*, Lecture Notes, Canadian Society for Civil Engineering, 1981
- 22 Fain, J., et al., *The ROSIE Language reference manual*, Technical Report N-1647-ARPA, Rand Corporation, Santa Monica, California 90406, 1981
- 23 Fenton, G.A., *Differential Movements and Stresses Arising in Masonry Veneers of High-rise Structures*, Master's Thesis, Department of Civil Engineering, Carleton University, Ottawa, April, 1984.
- 24 Fenton, V.C., *Limit States Design of Masonry. An Assessment of Proposed New Canadian Building Requirements*, Master's Thesis, Department of Civil Engineering, Carleton University, Ottawa, December, 1983.
- 25 Fiellhiem, R. and Syverson, P., *An Expert System for Sesam-69 Structural Analysis Program Selection*, Technical Report CP-83-6010, Division for Data Technology, Computas, Norway, 1983.
- 26 Forgey, C.L., *OPS5 User's Manual*, Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1981.
- 27 Gaschnig, J., "Prospector: An Expert System for Mineral Exploration", *Studies in Cybernetics: 1, Introductory Readings in Expert Systems*, D. Michie (ed.), pp. 47-64, Gordon and Breach, New York, 1982.
- 28 Gaschnig, J., Reboh, R. and Reiter, J., *Development of a Knowledge-Based Expert System for Water Resource Problems*, Technical Report SRI Project 1619, SRI International, August, 1981.
- 29 Hayes-Roth, F., Waterman, D.A. and Lenat, D.B. (eds.) *Building Expert Systems*, Addison-Wesley, 1982.
- 30 Holmes, M., "Steel Frames with Brickwork and Concrete Infilling", *Proceedings of the I.C.E.*, 19, pp. 473-478, 1961.
- 31 Holtz, N.M., *Symbolic Manipulation of Design Constraints*, Ph. D. Thesis, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, 1982.
- 32 Ishizuka, M., Fu, K.S., and Yao, J.T., *Inexact Inference for Rule-Based Damage Assessment of Existing Structures*, Technical Report CE-STR-81-5, Purdue University, February, 1981. (See also IJCAI81 pp. 837-842).

- [33] Ishizuka, M., Fu, K.S., and Yao, J.T., "Rule-based Damage Assessment of Existing Structures", *SM Archives* 8, pp. 99-118, Martinus Nijhoff Publishers, The Hague, 1983.
- [34] Lin, T.Y. and Stotesbury, S.D., *Structural Concepts for Architects and Engineers*, John Wiley and Sons, 1981.
- [35] Maher, M.L., Sriram, D., and Fenves, S.J., *Tools and Techniques for Knowledge-Based Expert Systems for Engineering Design*, Advances in Engineering Software, 1984.
- [36] Maher, M.L. and Fenves, S.J., *HI-RISE: A Knowledge-Based Expert System for the Preliminary Structural Design of High Rise Buildings*, Report No. R-85-146, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh 1985.
- [37] MacCallum, K.J., *Creative Ship Design by Computer. Computer Applications in the Automation of Shipyard Operation and Shipyard Design IV*, North Holland Publishing Company, 1982.
- [38] Martin, N., et al. "Knowledge-based Management for Experiment Planning in Molecular Genetics", *IJCAI77*, pp. 882-887, 1977.
- [39] Mendelson, E., *Introduction to Mathematical Logic*, Van Nostrand Reinhold, 1964.
- [40] McDermott, J., *R1: A Rule Based Configurer of Computer Systems*, Technical Report CMU-CS-80-119, Department of Computer Computer Science, Carnegie-Mellon University, Pittsburgh, 1980.
- [41] Mohamed, M.A., *Analysis of Low-Rise Masonry Walls Subjected to Thermal and Moisture Deformations*, Master's Thesis, Department of Civil Engineering, Carleton University, Ottawa, March, 1984.
- [42] Minsky, M., "A Framework for Representing Knowledge", *The Psychology of Computer Vision*, P. H. Winston (ed.), McGraw Hill, 1975.
- [43] National Building Code of Canada. *National Building Code of Canada 1980 and The Supplement to the National Building Code of Canada 1980*, NRCC No. 17303 and 17724, 1980.
- [44] National Building Code of Canada. *Supplement No. 1. Climatic Information for Building Design in Canada*, NRCC No. 15556, 1977.
- [45] National Research Council of Canada. *Cracks, Movements and Joints in Buildings*. Record of the Division of Building Research Building Science Seminar, Autumn 1972, Proceedings: no 2, NRCC 15477, September, 1976.
- [46] Negoita, C.V., *Expert Systems and Fuzzy Sets*, Benjamin/Cummings, 1985.
- [47] The New English Bible, Old Testament, Oxford University Press, Cambridge University Press, 1961.
- [48] Nii, H.P. and Aiello, N., "AGE (Attempt To Generalize): A Knowledge-based Program for Building Knowledge-based Programs", *IJCAI79*, pp. 645-655, 1979.
- [49] Nii, H.P. and Feigenbaum, E.A., "Rule-based Understanding of Signals" *Pattern Directed Inference Systems*, D. Waterman and F. Hayes-Roth (eds.), pp.483-501, Academic Press, New York, 1978.
- [50] Randall, F.A., and Panarese, W.C., *Concrete Masonry Handbook*, Portland Cement Association, Skokie, Illinois, 1976.
- [51] Reboh, R., *Knowledge Engineering Tools in the Prospector Environment*, Technical Report 8172, SRI International, June, 1981.

3 3
OF / DE

- [52] Rich, E., *Artificial Intelligence*, McGraw Hill, 1983.
- [53] Rychener, M.D., *PSRL: An SRL-Based Production-Rule System*, Technical Report DRC-005-03-84, Design Research Center, Carnegie-Mellon University, Pittsburgh, 1984.
- [54] Sahlin, S., "Interaction of Brick Masonry Walls and Concrete Slabs", *Designing, Engineering and Construction with Masonry Products*, E. B. Johnson (ed.), Gulf Publishing Company, pp. 266-277, 1969.
- [55] Sahlin, S., *Structural Masonry*, Prentice-Hall, 1971.
- [56] Schnieder, R.R. and Dickey, W.L., *Reinforced Masonry Design*, Prentice-Hall, 1980.
- [57] Shortliffe, E.H., *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York, 1976.
- [58] Simon, H. A., *The Sciences of the Artificial, 2 ed.*, The MIT Press, Cambridge, Massachusetts, 1980.
- [59] Smith, R.C., Honkala, T.L., and Andres, C.K., *MASONRY: Materials Design Construction*, Reston Publishing Co., Reston, Virginia, 1979.
- [60] Soame, A.J.M., "Interaction of Brickwork Walls and Concrete Floors Under Lateral Load", *Designing, Engineering and Construction with Masonry Products*, E. B. Johnson (ed.), Gulf Publishing Company, pp. 278-284, 1969.
- [61] Sorenson, C.P., and Tasker, H.E., *Cracking in Brick and Block Masonry*, Technical Study No. 43, Department of Works, Commonwealth Experimental Station, Australia, 1965.
- [62] Sriram, D., Maher, M.L., Bielak, J. and Fenves, S.J., *Expert Systems in Civil Engineering — A Survey*, Technical Report R-82-137, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, June, 1982.
- [63] Sriram, D., *A Bibliography on Knowledge-Based Expert Systems in Engineering*, Technical Report DRC-12-23-84, Design Research Center, Carnegie-Mellon University, Pittsburgh, 1984.
- [64] Sriram, D. and Fenves, S.J., "Destiny: A Knowledge-Based Approach to Integrated Structural Design" Prepared for presentation at the Symposium on Artificial Intelligence, École Polytechnique, Montreal, Canada, November 9, 1984.
- [65] Stafford-Smith, B., "Methods for Predicting the Lateral Stiffness and Strength of Multi-storey Infilled Frames", *Building Science*, 2, 1967.
- [66] Stallman, R. and Sussman, G.J., "Forward Reasoning and Dependency-directed Backtracking in a System for Computer aided Circuit Analysis", *Artificial Intelligence*, 9, pp. 135-196, 1977.
- [67] Stanford, G., *Potential Applications of Expert Systems in Geotechnical Engineering*, Master's Thesis, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, April, 1983.
- [68] Steele, G.L., *The Definition and Implementation of a Computer Programming Language Based on Constraints*, Technical Report AI-TR-595, MIT AI Laboratory, Cambridge, Massachusetts, 1980.
- [69] Stefik, M., *Planning with Constraints*, Technical Report STAN-CS-80-784, Computer Science Department, Stanford University, California, January, 1980.
- [70] Sussman, G.J. and Steele, G.L., "CONSTRAINTS — A Language for Expressing Almost Hierarchical Descriptions", *Artificial Intelligence*, 14, pp. 1-39, 1980.

- [71] Turing, A.M., "Computing Machinery and Intelligence," *Mind*, Vol. LIX, No. 236, 1950.
- [72] Van Melle, W., "A Domain-independent Production System for Consultation Programs", *IJCAI79*, pp. 923-925, 1979.
- [73] Wiess, S. and Kulikowski, C., "Expert: A System for Developing Consultation Models", *IJCAI79*, pp. 942-947, 1979.
- [74] Winston, P.H., *Artificial Intelligence*, Addison-Wesley, 1977.
- [75] Winston, P.H., and Horn, B.K.P., *LISP*, Addison-Wesley, 1981.
- [76] Wright, J. M. and Fox, M., *SRL: Schema Representation Language*, Robotics Institute Technical Report, Carnegie-Mellon University, Pittsburgh, 1983.

END

1 6 0 4 8 6

FIN