EXPERT SYSTEMS FOR FAULT ANALYSIS

STEPHEN HOWARD BOOTH

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

July 1993

The University of Aston in Birmingham

Expert Systems for Fault Analysis

Stephen Howard Booth

PhD

SUMMARY

The initial aim of this research was to investigate the application of Expert Systems, or Knowledge Base Systems technology to the automated synthesis of Hazard and Operability Studies (HAZOP). Due to the generic nature of fault analysis problems and the way in which Knowledge Base Systems work, this goal has evolved into a consideration of automated support for fault analysis in general, covering HAZOP, Fault Tree Analysis, FMEA and Fault Diagnosis in the process industries.

This thesis describes a proposed architecture for such an Expert System. The purpose of the system is to produce a descriptive model of faults and fault propagation from a description of the physical structure of the plant. From these descriptive models, the desired fault analysis may be produced.

The way in which this is done reflects the complexity of the problem which, in principle, encompasses the whole of the discipline of process engineering. An attempt is made to incorporate the perceived method that an expert uses to solve the problem; keywords, heuristics and guidelines from techniques such as HAZOP and Fault Tree Synthesis are used.

In a truly Expert System, the performance of the system is strongly dependent on the high quality of the knowledge that is incorporated. This expert knowledge often takes the form of heuristics or rules of thumb which are used in problem solving. This research has shown that, for the application of fault analysis heuristics, it is also necessary to have a representation of the details of fault propagation within a process. This depth of knowledge helps to ensure the robustness of the system - a gradual rather than abrupt degradation at the boundaries of the domain.

Keywords : Expert Systems, Knowledge Base Systems, Fault Analysis, HAZOP, Fault Tree Synthesis

Acknowledgements

The author would like to acknowledge the valued support of the following people:

SERC for funding the research.

Mike Jones - for being the initial supervisor.

John Fletcher - for being the ultimate supervisor.

Sue Whalley for the endurance record.

David Lihou and colleagues at Aston for helping to get this research off the ground.

The UGC for attempting to bury it again.

CONTENTS

Page

SUMMARY **3**

CHAPTER ONE: INTRODUCTION **14**

1.1 RESEARCH AIMS 14
1.2 FAULT ANALYSIS AND KNOWLEDGE BASE SYSTEMS 14
1.3 RESEARCH METHOD 16
1.4 THESIS OVERVIEW 17

CHAPTER TWO: LITERATURE REVIEW
- Safety and Reliability **19**

2.1 INTRODUCTION 19
2.2 SYSTEM RELIABILITY - An Historical Perspective 19
2.3 SYSTEM SAFETY AND RELIABILITY 33
2.3.1 Literature Overview 33
2.3.2 PRA / HAZAN 40
2.4 HAZARD AND OPERABILITY STUDIES 42
2.4.1 Background 42
2.4.2 Application of HAZOP 42
2.4.3 HAZOP Procedure 43
2.4.4 Difficulties with HAZOP 46
2.4.5 Recent Developments 47
2.5 FAULT TREE ANALYSIS 54
2.5.1 Background 54
2.5.2 Fault Tree Analysis 55
2.5.3 Fault Tree Synthesis 57
2.5.4 Automated Fault Tree Synthesis 62
2.5.5 Fault Tree Evaluation 79
2.6 OTHER CAUSAL REPRESENTATIONS 81

	Page
2.7 FAULT DIAGNOSIS	84
2.7.1 Introduction	84
2.7.2 Outline of Computer Models	85
2.7.3 Causal Models	87
2.7.4 Structural Models	89
2.7.5 Discussion	91
CHAPTER THREE: LITERATURE REVIEW - Knowledge Based Systems	**93**
3.1 INTRODUCTION	93
3.2 AN HISTORICAL PERSPECTIVE	93
3.3 LITERATURE OVERVIEW	97
3.4 COMPUTERS AND SOFTWARE	102
3.4.1 Computer Hardware	102
3.4.2 Knowledge Based Systems Software	103
3.5 KNOWLEDGE BASED SYSTEMS	108
3.5.1 Introduction	108
3.5.2 Knowledge Representation	110
3.5.3 Knowledge Processing	122
3.5.4 Knowledge Acquisition	125
3.6 ARTIFICIAL INTELLIGENCE	127
3.6.1 Introduction	127
3.6.2 Reasoning About Physical Systems	128
CHAPTER FOUR: METHOD	**134**
4.1 HARDWARE	134
4.2 SOFTWARE	135
4.3 GOALS OF THE RESEARCH	135
4.4 EXPERT SYSTEM DESIGN METHODOLOGY	136
CHAPTER FIVE: IKBS DESIGN EVOLUTION	**138**
5.1 INTRODUCTION	138
5.2 FAULT ANALYSIS	142
5.2.1 Fault Analysis Knowledge	142
5.2.2 Manual Fault Analyses	145

Page

5.3 THE REPRESENTATION OF PROCESS PLANT 148
5.3.1 Introduction 148
5.3.2 Alternative Representations 148
5.3.3 Stereotypes 151
5.4 CAUSAL REPRESENTATION AND KNOWLEDGE 155
5.4.1 The Representation of Causality 155
5.4.2 Causal Knowledge 155
5.4.3 Production of the Fault Analysis 159
5.5 DISCUSSION 160
5.5.1 Levels of Abstraction 160
5.5.2 Symbolic Modelling 161
5.5.3 The Goals of the Research. 163

CHAPTER SIX: THE PROPOSED DESIGN 165
6.1 INTRODUCTION 165
6.2 OVERVIEW OF PROPOSED ARCHITECTURE 166
6.3 STRUCTURE - REPRESENTATION AND ENHANCEMENT 169
6.3.1 Introduction 169
6.3.2 Representation of Process Plant Structure 170
6.3.3 Initial Structure Representation 176
6.3.4 Final Structure Representation 180
6.4 EVENTS - REPRESENTATION AND DEFINITION 183
6.4.1 Introduction 183
6.4.2 Representation of Events 183
6.4.3 Definition of Events 186
6.5 CAUSALITY - REPRESENTATION AND REASONING 189
6.5.1 Introduction 189
6.5.2 Representation of Causality 190
6.5.2 Definition of Causal Relationships 191
6.6 PRODUCTION OF THE FAULT ANALYSIS 193
6.6.1 Introduction 193
6.6.2 HAZOP using Cause and Symptom Equations 193
6.6.3 Fault Tree Synthesis 195
6.6.5 Other Fault Analyses 195

	Page
CHAPTER SEVEN: DISCUSSION	**197**
7.1 INTRODUCTION	197
7.2 DESIGN EVALUATION	197
7.2.1 Introduction	197
7.2.2 The Architecture	198
7.2.3 The Knowledge	201
7.2.4 The Mechanics of Experimentation	201
7.2.5 Evaluating the Success of the Design	205
7.3 GENERALIZATION	206
7.3.1 Generalizing to Other Applications	206
7.3.2 Constructing an Operational KBS	209
CHAPTER EIGHT: CONCLUSION	**212**
REFERENCES	**215**
APPENDICES	**248**
Appendix 1 : Ammonia Let Down Case Study Paper	249
Appendix 2 : Equipment Failure Mode Tables	262
Appendix 3 : Example Case Study Knowledge Base Rules	265

FIGURES

FIGURE	TITLE	Page
Chapter Two		
2.3.1a	Outline of Literature Classification	33
2.3.1b	System Safety and Reliability - SSR	34
2.3.1c	Risk Analysis - RA	35
2.3.1d	Causal Analysis - CA	36
2.3.1e	Fault Tree Synthesis - FTS	37
2.3.1f	Fault Diagnosis - FD	38
2.3.1g	Human Factors - HF	39
2.3.2a	Risk Assessment	41
2.4.3a	Detailed Sequence of an HAZOP Study	45
2.4.5a	Ammonia Let-Down Two Phase Splitter	48
2.4.5b	An Example of a Cause and Symptom Equation Rule	53
2.5.2a	A Simple Fault Tree	55
2.5.3a	Causal Event Classification	59
2.5.3b	Component Failure Characteristics	60
2.5.4a	Component Failure Transfer Function	65
2.5.4b	Decision Table and Mini-Fault Trees - a simple pump	69
2.5.4c	Example Component Variable Digraph	71
2.5.4d	Plant Digraph for variable T4 of the Nitric Acid Cooler	72
2.5.4e	Generalised Fault Tree for a Feedback Control Loop	73
2.7.3a	Encoding a Fault Tree as a Rule	88

FIGURE	TITLE	Page
Chapter Three		
3.3a	Expert Systems - ES	98
3.3b	Intelligent Knowledge Base Systems - IKBS	99
3.3c	Artificial Intelligence - AI	100
3.3d	Applications - APP	101
3.5.2a	Production Rules and their Tree Representation	115
3.5.2b	Architecture of a Simple Rule Based System	116
3.5.2c	Search Mechanisms	118
3.5.2d	A Simple Semantic Net	119
3.5.2e	Frames	120
Chapter Four		
4.4a	A Flowsheet for Expert System Design	137
Chapter Five		
5.1a	Initial Conception of the Solution to the Problem	138
5.1b	Overview of the Design Evolution	140
5.2.1a	Fault Analysis Knowledge	142
5.3.2a	Two Phase Splitter P&ID	149
5.3.2b	Examples of Early Frames	149
5.3.2c	Hierarchical Representation of the Two Phase Splitter	150
5.3.3a	Type / Identity Stereotypes	151
5.3.3b	Classification of Stereotypes by Identity	152
5.3.3c	Hierarchical Classification of Process Plant Objects	154
5.4.2a	Rules Deriving Causality from Structure	156
5.4.2b	Overview of Causal Analysis Production	158
5.5.2a	Partial Model of a Transmitter	161

FIGURE	TITLE	Page
Chapter Six		
6.2a	The Proposed Architecture	167
6.3.1a	Overview of the Process of Structural Representation	169
6.3.2a	Two Phase Splitter - P+ID and Graphical Representation	172
6.3.2b	The is-a Hierarchy of Objects	173
6.3.2c	Frame for LCV1	174
6.3.3a	Two Phase Splitter - Initial Objects	176
6.3.3b	Initial Syntax for Nodes	178
6.3.3c	Initial Stereotype and Instantiation for LCV1	179
6.3.4a	Final Frame for LCV1	182
6.4.2a	Event Types and Syntax	184
6.4.3a	Definition of Events from Structure Representation	186
6.4.3b	Generic Event Definitions for Devices	187
6.4.3c	Example Event Definition Rule	188
6.5.1a	Causal Definition in the KBS	189
6.6.1a	Production of the Causal Analysis	193
6.6.2a	Production of Cause and Symptom Equations	194
Chapter Seven		
7.2.4a	Frame for LT1	202
7.2.4b	General Form of a Rule	203
7.2.4c	Example Rule	203
7.3.1a	Syngas Desorber - Graphical Representation	207
7.3.1b	Hydrocarbon Reactor - Graphical Representation	208

TABLES

TABLE	TITLE	PAGE
Chapter Two		
2.4.2a	Problems Revealed by Analysis of the Results from Ten HAZOPs	43
2.4.3a	HAZOP Guide Words	46
2.4.5a	Property Words	49
2.4.5b	Index Definitions for Events in the Ammonia Let-Down System	49
2.5.3a	Fault Tree Event Symbols	58
2.5.3b	Fault Tree Logic Gate Symbols	58
Chapter Six		
6.3.3a	Stereotype Slot Values	179

CHAPTER ONE

INTRODUCTION

1.1 RESEARCH AIMS

This research arose from the possibility of using rules for assistance in Hazard and Operability Studies (HAZOP) recorded using logical equations [Lihou 1982d]. The initially defined goal of the research was to attempt to produce these equations automatically, using a Knowledge Base System (KBS), sometimes referred to as an Expert System. For a number of reasons, involving both the problems of HAZOP synthesis and the way in which KBS operate, this initial goal has evolved into a consideration of Knowledge Base System support for fault analysis in general. This involves the computerised provision of assistance for Fault Tree Synthesis, HAZOP, Fault Diagnosis and other similar techniques from a physical description of the chemical process under consideration.

1.2 FAULT ANALYSIS AND KNOWLEDGE BASE SYSTEMS

1.2.1 **Fault Analysis**

As chemical plant has become larger and more complex, handling increasingly dangerous materials, the potential hazards have increased correspondingly. The problems of maintaining safety and reliability of both technology and its human factor have also become crucial to the success of the nuclear, military, electronics and aerospace industries. Essentially the problem has become one of Risk Assessment - determining the causes and likelihood of mission or system failure and combining this with a measure of the consequences, be they hazards or operational problems. Central to Risk Assessment, therefore, is the understanding of the propagation of faults within a system, for identification or causal assessment of undesirable incidents. Fault propagation is also central to Fault Diagnosis, Alarm Analysis and control of equipment or operations.

Analysis techniques such as HAZOP or Fault Tree Analysis require a large expert manpower investment; this implies that any automated assistance to the analysis could prove economically valuable. Fault Diagnosis or the analysis of alarms in hazardous installations such as nuclear power stations is obviously a critical task; any Decision Support System capable of assisting in the rapid and accurate diagnosis and mitigation of faults is of clear value.

1.2.2 **Knowledge Base Systems**

The initial conception of Artificial Intelligence (AI) in the late 1950s was as a means of understanding intelligence by attempting to reproduce it. These attempts involve the emulation of human activities such as natural language understanding and synthesis, visual perception, problem solving and game playing.

Central to most AI work is the use of symbol structures rather than conventional data structures; symbolic processing languages such as LISP are used rather than data processing languages such as FORTRAN. Some AI work involves the explicit symbolic representation of human knowledge. It became apparent, around 1970, that the more successful problem solving systems could be characterised largely by the high quality of the knowledge that they explicitly represented.

This concentration on the symbolic representation of knowledge led to Knowledge Base or "Expert" Systems as they are known today. There are currently a number of highly successful KBS and a plethora of smaller applications in areas as diverse as medicine, mineral prospecting, economic modelling, law, tax evasion, engineering design and mycology.

1.2.3 **Fault Analysis Using Knowledge Base Systems**

In the 1970s considerable efforts were devoted to computer programs for the automated synthesis of fault trees, but none of these have found widespread application. In more recent years there has been considerable interest in systems for the automated provision of decision support for Fault Diagnosis, but most of these systems rely on an existing fault analysis conducted in the usual manner. There is clearly a need for producing fault analyses from a structural description of the analysed system.

In conventional HAZOP, analysts make use of key words or symbols. The results of techniques such as Fault Tree Analysis are readily expressed and qualitatively analysed using symbol-structures. For both these techniques, workers have developed structured procedures, rules and guidelines to help produce understandable and correct analyses. Fault Analysis and Diagnosis is often the province of experts who reason using heuristics or rules of thumb in preference to underlying theoretical principles.

These characteristics of the task of fault analysis, principally its symbolic nature, and the apparent lack of success of conventional approaches suggested the application of KBS to the problem. However, reliance on automated synthesis is unlikely to be seen in the near future. There are a number of reasons for this, not the least of which are the complexity of the problem to be solved and the need for a high degree of confidence in safety matters.

1.3 RESEARCH METHOD

Development of an Expert, or Knowledge Base, System is often characterised by an iterative process of refinement or, in some cases, starting over. The initial aim of the iterative process is to develop a prototype KBS which is capable of solving a particular representative problem of the area of application - the domain. This prototype will then hopefully embody an appropriate means of representing and manipulating domain knowledge and will include concepts and ontologies which characterise the knowledge. The next step is the generalization of the prototype to the domain as a whole; the success or failure of this extension determines whether another iteration is necessary.

This is the research methodology which has been adopted here. A case study was selected and the body of the research was concerned with the development of a prototype system that would solve the case study and also provide a suitable architecture for a full application system. It did not prove possible to obtain the necessary facilities for the development of a fully operational system. Key elements of the design were evaluated using, principally, COMMON LISP implemented on an IBM PC/AT and later on a University mainframe computer.

1.4 THESIS OVERVIEW

This thesis is concerned with two disciplines, that of fault analysis as a part of chemical process safety and reliability, and that of Intelligent Knowledge Base Systems (IKBS) - a technology arising from Artificial Intelligence (AI) research. The thesis commences, therefore, with literature reviews dealing with these disciplines.

The first literature review discusses safety and reliability. It begins with an historical perspective and an overview of the context of fault analysis within loss prevention and risk assessment. A hierarchical classification of pertinent literature is presented. The bulk of the review is concerned with three principal areas of fault propagation: HAZOP, Fault Tree Analysis and Fault Diagnosis, and focuses on synthesis procedures and research within these areas. The chapter closes with a brief review of future prospects.

The second literature review commences with a brief discussion of the history of AI and the emergence of Knowledge Base, or Expert, Systems technology. A hierarchical classification of pertinent literature is presented. Knowledge Base Systems involve unconventional computer systems which operate using symbol manipulation rather than data processing; the hardware and software requirements and options for these systems are described. The bulk of the review is concerned with techniques for the representation and application of knowledge by computer. A section on Artificial Intelligence discusses the research which underpins the technology of IKBS, focusing on that which is concerned with the modelling of problem solving in physical systems. The chapter concludes with a brief discussion of the application of IKBS and the future prospects.

The central topic of the thesis is the design of an IKBS for assistance in Fault Analysis. Chapter Four discusses the method behind the design whilst Chapters Five and Six discuss the evolution of the design and the design itself.

Chapter Five covers the research and experimentation that contributed to the design. This involves the detailed consideration of fault analysis knowledge and procedures in an attempt to structure this information to facilitate symbolic representation and reasoning. The representation, in a computer, of the process plant under consideration is clearly of major importance and a key section is devoted to this topic.

Chapter Six presents an overview of the proposed IKBS Design and then discusses each of the stages of the fault analysis process as the design is put into practice: the representation of process plant structure; the production of cause-effect knowledge from the plant structure; and the production of the fault analysis. Expert Systems are usually designed using a representative case study and here all facets of the design are described as they relate to the selected case study - an Ammonia Let-Down System HAZOP.

Chapter Seven discusses the proposed design and its evaluation and discusses how the principles of the design might be used in a fully fledged application system with a view to solving a range of problems. Chapter Eight summarises the key parts and conclusions of the thesis and presents a suggested future research plan.

CHAPTER 2

LITERATURE REVIEW - SAFETY AND RELIABILITY

2.1 INTRODUCTION

This literature review discusses the state of the art with respect to the synthesis of fault analysis using Intelligent Knowledge Based Systems. This is the generic problem area which encompasses the original aim of this research: IKBS synthesis of Hazard and Operability Studies. The discussion is presented in the context of system safety and reliability as a whole and focuses on the techniques of HAZOP, Fault Tree Analysis and Fault Diagnosis. The Chapter opens with an historical review of Safety and Reliability Assessment.

2.2 SYSTEM RELIABILITY - AN HISTORICAL PERSPECTIVE

2.2.1 Introduction

Reliability has played a part in people's lives since the dawn of technology in the stone age. In the art of stone-tool making it was soon recognised that flint gave a sharper cutting edge than other available materials. Some veins of flint produced longer-lasting edges than others. The relative longevity of flint tools implies a higher degree of reliability according to the reliability engineer's definition [Green 1976]:

> "That characteristic of an item expressed by the probability that it will perform its required function in the desired manner under all the relevant conditions and on the occasions or during the time intervals when it is required so to perform."

It is interesting to consider the process of the selection of flint above other materials. The selection was made entirely on the basis of experience; experimenting with different rocks revealed the best choice. This is the essence of empiricism - qualitative assessments founded on experience and observation.

The empirical method sustained technological development through the industrial revolution and up to the beginning of the present century. Medieval cathedrals were built according to the principles established in tried and tested designs, the most reliable of which were indicated by the longest lasting structures; few structural analysis techniques were available. In pre-20th century construction, one device used to ensure reliability or longevity is sometimes immediately apparent - overdesign. The massive ironworks of Tower Bridge in London, for example, are surely not entirely necessary for structural integrity. This practice of overdesign to ensure the ability to meet the intended purpose (or to cater for unanticipated requirements) continues to the present day; process vessels are commonly oversized by 10-20%.

The scientific revolution that stemmed from the Renaissance brought a new approach to industry: the application of analysis and the development of scientific theories to explain the physical world. The sheer complexity of machinery wrought in the industrial revolution demanded some ability to predict the behaviour of theoretical systems in order to assess their suitability for their intended purpose.

The rapidity of industrial development in the early 20th century, particularly in the military sphere, produced machine systems whose successful operation depended on a large number of failure-susceptible components. This required the ability to optimise the longevity of the intended purpose i.e. the reliability of the system. The qualitative or empirical approach was no longer economically or environmentally acceptable. An unreliable design was revealed by operating problems or as a result of hazardous incidents, the "every dog gets one bite" approach. Processing of petrochemicals and toxic materials on a large scale meant that the bites of the dogs were becoming very serious indeed. The problem of <u>prevention</u> of "dog bites" is particularly pertinent in the nuclear, aerospace and military spheres.

The requirement to be able to predict systems reliability in a quantitative manner gave rise to reliability analysis as it is known today. Analytical techniques are used to express reliability either relatively, or in terms of temporal probabilities. These data may then be used to express the availability of a system or the risk associated with a particular enterprise, the process providing valuable insights into the operability and reliability implications of a given design.

The evolution of reliability assessment, then, can be seen to parallel that of technology as a whole: the gradual transition from a largely empirical art to a largely analytic science.

2.2.2 Systems Reliability in the Twentieth Century

The modern chemical processing industry has its roots in the 19th century alkali manufacturers who served the booming UK textile industry. Production of small scale, high purity chemicals, particularly dyestuffs, gave rise to the fine chemicals industries which, in the early 20th century, served the major protagonists in World War I. With the discovery and exploitation of crude oil, and the subsequent development of large-scale refining came the concept of Unit Operations in the US in the 20s; a means of structuring processing operations. At this time process control was entirely manual, large buffers and surge tanks were used to regulate process fluctuations.

The military necessities of World War II produced rapid technological advances. The ingenuity of German engineering produced military machines of great potential effectiveness but unfortunate complexity and poor reliability. This contrasted sharply with the Soviet strategy of producing large numbers of cheap, simple, reliable and easily maintained machines - evidenced by the success of the T-34 tank. (This military contrast can be seen today, although to a lesser extent: Western sophistication and expense versus Soviet simplicity, cheapness and large numbers.)

Servicing and maintaining military machinery gave rise to severe logistics problems. Particularly in the US, efforts were made to tackle these problems by emphasizing the requirement for components of maximum performance requiring minimum aftercare. The failure of the Wehrmacht to learn this lesson culminated in the failure of the Ardennes offensive in 1944, when tank divisions ground to a halt for want of logistic support.

The intense scientific activity surrounding the V1 missile programme, however, brought together engineering judgement and statistics to produce the Lusser product law of reliability [Henley 1981]:

$$R_{system} = R_1 * R_2 * R_3 \ldots$$

Where Ri are the reliabilities of subsystems each of which is necessary for overall success.

The implication that a single weak link could be more reliable than a number of relatively strong links led to great improvements in the mission success rate of the V1.

2.2.3 Post - WWII

After the war the US Army continued economic studies of the operation and maintenance of military vehicles; the Army and subsequently the automobile industry were able to use these studies in defining optimum maintenance and replacement intervals. It was during this period that the discipline of ergonomics was born out of the necessity to design military systems according to the needs and parameters of the human operators. The embryo electronics industry was beginning to consider the reliability of thermionic valve systems with a view to establishing acceptable standards, whilst at the same time considering the implications of the discovery of the transistor in 1947. Following the deployment of nuclear weapons at the end of World War II, nuclear power programmes were instigated in the context of the enormous destructive potential inherent in nuclear materials. The horrific consequences of human exposure to nuclear radiation at Nagasaki and Hiroshima were becoming known, although they were suppressed for political reasons to allow atmospheric nuclear testing in conditions which, with hindsight, can be seen to have had serious consequences.

1950

In the early 1950s these hazards raised an awareness of the need to be able to predict the likelihood and the consequences of nuclear accidents. In 1957 the first published study of such matters was produced as WASH-740. This report concentrated on the "worst credible incident" (WCI) that could occur as a result of a specific accident sequence. The WCI was accepted to be a large loss-of-coolant-accident (LOCA) leading to core degradation and meltdown, breach of containment and contamination of the environment by nuclear material. Although there was some discussion of incident likelihood and of acceptable upper bounds for the frequency of occurrence, no attempt was made to produce quantification for any specific accident scenario. This raised the question of the "how" of the specific "what" postulated.

The result of the failure of consequence limiting equipment was made clear by the release of nuclear material at Windscale in 1957 which necessitated the disposal of large quantities of local milk followed by an extensive clean-up programme.

In the process industries, economics and the realities of competition became the driving force in improving the efficiency of increasingly complex operations. Basic feedback loops were evolving into complex automatic control and safety systems. The availability of data relating to the inherent hazards of process materials (combustibility, toxicity etc) enabled the Dow Chemical Company to produce its first Fire and Explosion Index in 1956 [Dow 1981]. This provided a means of Hazard Ranking according to materials quantity and quality along with simple operational parameters. The resulting numerical index could then be used to relate the potential hazards of a new design to those of an existing facility. This permitted an assessment of hazard acceptability in the context of existing hazard levels.

In the 1950s the electronics industry demonstrated a high standard of reliability of computers constructed entirely from transistors. Evolution of component testing methods allowed the establishment of standards for electronic components. Resistor tolerances were standardised in 1957 in terms of allowable percentage deviation from the stated value (commonly 20,10,5,2 and 1%). This required an understanding of the capabilities and limitations of the manufacturing processes. [Coppola 1984]

A perennial problem in reliability analysis is the collection and interpretation of failure data. In comparison with other physical systems, electronic components are small and discrete and therefore more readily defined. A large number of identical components which are situated in a controlled, usually non-hostile environment provide a good source of coherent failure data. Electronic component failure modes and system failure models are also more easily defined; this is particularly true of digital systems. Functional testing, accelerated life testing and structured fault diagnosis are also more readily carried out. It is for these reasons that many advances in the collection and interpretation of failure data and in the development of reliability theory have been made in the field of electronics; the techniques derived have then found wider application in physical systems engineering.

It was in the early 1950s that the distinction between overall equipment failure rates and the breakdown into failure mode rates was made. Each distinct component failure mode has a different implication for the success of the system as a whole. Efforts were also being made to develop mathematical models to represent component failure distributions. The simple yet highly versatile exponential distribution for random failure was extended to become the Weibull distribution [Lihou 1982a], in which parameters are used to define an appropriate distribution to correlate with experimental data. Markov processes were used to model systems in terms of state space transformations, in which the various states of a system along with transition paths are described topologically [Joller 1982a]. In the early 1960s Monte Carlo methods came into use as a means of assessment using random sampling, and of modelling using random stimulation.

1960

The 1960s saw the US, UK and French nuclear power programmes well under way. Safety considerations still focused on the worst credible case with the corresponding unfortunate influence on public opinion. In contrast Farmer [Fussell 1977] emphasised the importance of ensuring the reliability of consequence limiting equipment. The space race saw a reliance on failure tolerant systems in environments where mission failure permits no possibility of recovery.

WASH-740 focused on the "what" of nuclear incidents. It became necessary to develop techniques to elucidate the "how" of such incidents, to enable quantitative assessments of risk to be made. One such fault modelling technique, Fault Tree Analysis (FTA), was developed by H A Watson of Bell Laboratories in 1961 to aid in the evaluation of the launch control system of the Minuteman missile. In this technique, the system failure event of interest is identified and the causal structure leading to this event is described diagrammatically in a tree of events connected by Boolean operators (logic gates). Following a safety symposium in 1965 when a number of papers were published on the subject [Haasl 1965] the technique gained widespread acceptance towards the end of the decade.

In the process industries the identification of this undesirable "top event" still presented problems, relying entirely on experiential judgement. The increased complexity of new designs gave rise to a need for structured techniques of Hazard Identification (HAZID). The use of HAZID checklists predated the technique of Failure Modes and Effects Analysis (FMEA), in which credible failure modes of components or sub-systems are identified and all important consequences of these failures are elucidated [Recht 1966a]. Once a major hazard is identified, consequences can be estimated, for example the effects of a release and subsequent ignition of flammable vapour. Newly developed mathematical models of fire, explosion and toxic effects facilitated this consequence assessment [ISGRA 1985].

Quantitative assessment of system reliability requires the availability of component and systems failure data. Moreover, this failure data must be in the appropriate form for the methods in which it is to be used. In the process industries in particular, a given component may be required to operate under a wide variety of conditions. In the consideration of a similar component in a newly-designed system, interpretation of gathered failure data must take account of these differences in the context of the expected operating environment of the new component. Although some companies maintained in-house data stores the majority of these data related to failures of specific components under specific conditions - of limited applicability to new designs. There was clearly a need for generalised, large-scale and widely accessible data sources. One of the first centralised reliability data banks to become available was the Electronic Equipment Reliability Data Store. The decade saw the establishment of a number of such databases, one of the most comprehensive of which became the UKAEA SYREL data bank of events and generic failure rates which continues in use today [Anon. 1977].

1970

By 1970 the semiconductor revolution was well under way. Texas Instruments had succeeded in putting many transistors on a single wafer of silicon substrate to produce the first integrated circuit (IC). The increasing scale of integration, proceeding rapidly by orders of magnitude, led to the invention of the microprocessor which resulted in the proliferation of digital computers in the 1970s. Chip manufacturers have production reliability problems of their own. A single wafer of silicon may have hundreds of chips etched onto it; the wafer is then sliced and each chip tested individually. In VLSI production, rejection rates of above 90% are not uncommon. These problems have been reduced by improvements in manufacturing techniques along with dramatic falls in production costs which have seen the widespread availability of cheap electronic devices in the 1980s.

Digital systems have an important advantage to the reliability analyst; their behaviour is strictly deterministic. System information consists of logical 0 or 1 states and the propagation of information is strictly prescribed by the system design. System behaviour, therefore, is readily modelled; in simple terms faults may be diagnosed as finding a 1 where a 0 is expected, or vice-versa. Digital components (eg logic gates) are perfectly discrete and similarly have simple failure modes. For these reasons computerised design, modelling and reliability analysis of digital systems led the way in the development of reliability theory, dynamic analysis and control theory in the early 1970s.

In the USA the proliferation of potentially hazardous nuclear sites and increasing public concern over nuclear safety stimulated reliability studies. These culminated in the US Nuclear Regulatory Commission report WASH-1400 [NUREG 1975].

WASH-1400 was the most comprehensive and in-depth study of reactor safety ever undertaken and represented a milestone in the development of probabilistic risk assessment (PRA). WASH-1400 was an attempt to calculate the risk from the operation of 100 light-water reactors in the USA. The principal causal modelling technique used was fault tree analysis, in which the probabilities of occurrence of hazardous events were evaluated with respect to expected component and sub-system failure rates. Solution of these fault trees, of which many were produced, was facilitated by computer codes which had become available in recent years [Lee 1985]. In conjunction with analyses of the outcomes of hazardous events, WASH-1400 attempted to evaluate the comparative risks associated with different incident scenarios. One result of this was the realization that the WCI of WASH-740 did not in fact present the largest risk. A number of high probability events of low consequences can present more of a risk than a low probability catastrophic event. The figures were dominated by numerous small LOCAs , rather than the major LOCA and subsequent core melt-down sequence of WASH-740.

WASH-1400 also emphasised the importance of human reliability considerations in evaluating overall system safety. This was illustrated dramatically in the Three Mile Island (TMI) incident in 1979. In TMI human misinterpretation of the situation resulting from mechanical failure led to the release of contaminated gas to the environment. The incident called into question the methods and conclusions of the WASH-1400 report. After initial criticism however, it was realised that WASH-1400 successfully predicted the early event sequences in TMI; the complete accident sequence was not predicted due to design differences in the TMI plant. The report of the President's Commission on TMI (the Kemeny Report) concluded that the WASH-1400 methodology was on the right track but lacked refinement and application. In the following years a number of PRA studies were published and PRA has since seen widespread application in many industries.

In the process industries emphasis was placed not only on safety but also on the economic aspects of operability and availability. Elucidation of potential failures and their consequences relies heavily on the general experience and specific knowledge of engineers and line managers. Large companies were conducting extensive safety audits and studies accompanying all stages from initial design formulation through to plant commissioning. At ICI in the early 1970s, refinement of the Method Study led to the development of Hazard and Operability Studies - HAZOP, to form an integral part of a Hazard Analysis - HAZAN (similar in concept to PRA) [Kletz 1985]. HAZOP addresses the problem of hazard identification in that causes and consequences are derived for an exhaustive list of all possible deviations from normal operating conditions and all possible equipment failure modes. It has since been adopted as a standard technique by many companies. It may be utilised not only for hazard evaluation but also for instrumentation and control design, maintenance planning and fault diagnosis. One disadvantage of HAZOP is that it requires a large investment of the time of experienced personnel. The resultant expense can easily exceed 1% of capital cost [Whitehouse

1982]. Lihou [1980c] developed a technique of recording events in the form of logical equations expressing the causes and consequences of process deviations and equipment failures. The technique has been shown to be considerably quicker than conventional tabular HAZOP. The cause and symptom equations may also be used to produce fault trees directly, thereby permitting probabilistic assessment. This represents an important bridge between FTA and HAZOP. FTA of chemical plant has been hindered by the problem of hazard identification as it is a "top down" technique requiring the prior identification of hazardous events.

In 1974 the town of Flixborough was shaken by the explosion of a cloud of cyclohexane vapour at the NYPRO UK chemical plant. Although the ultimate cause of the accident was mechanical - the failure of a by-pass pipe - the pipe had been installed as a temporary measure and was operated in unsafe conditions for a longer period than originally intended. The incident illustrated the role that the failure of management systems could play; managers are not always cognizant of the implied hazards. Following public outcry the Health and Safety Executive (HSE) compiled a list of 1700 hazardous sites in the UK. Surprisingly this was the first time that this had been done. Calls for tighter control by national and local government brought about legislation and planning controls in subsequent years. Following the Seveso disaster in Italy in 1976, in which a cloud of highly toxic dioxin contaminated gas was released, the EEC has taken steps to tighten safety measures.

For reliability analysts concerned with fault propagation, Fault Tree Analysis (FTA) dominated the 1970s. In view of the considerable resources necessary to produce comprehensive fault trees attempts have been made to computerise this process. The first published attempt at this was Fussell's Synthetic Tree Method (STM) [Fussell 1973b] applied to electronic systems. In essence the method consists of decomposing the system into individual components, for each of which a mini fault tree exists in a library. The minitrees are then combined to produce the overall fault tree. Although the method has its limitations it marked the beginning of a more widespread effort towards automated Fault Tree Synthesis (FTS). These methods, largely based on causal models of system components have been applied to electronic, nuclear and chemical processing systems.

The first attempt at automated FTS for a chemical plant was made by Lapp and Powers [Lapp 1976]. Their approach was novel in that the synthesis was founded on plant digraphs - a representation of the nature of the interrelationships between process variables. The original paper was followed by a rather puzzling bout of frenzied nitpicking in the literature, concerned with a possible logical flaw brought about by Lapp and Powers' use of XOR gates. By 1980 progress in this area had been severely limited by the problems of obtaining logically credible fault propagation analyses . By this time other methods of representing fault propagation had been found, these included Cause Consequence Diagrams and Event Trees .

1980

In recent years economics have continued to be a driving force in the pursuit of increased safety and reliability. An apparent financial crisis in 1984 brought the US nuclear power industry to a virtual standstill; no new power station projects have been initiated in this decade. In the UK the long lasting Sizewell B public enquiry was symptomatic of this trend. The reason for the financial problems is that safety considerations have now become integrated into the costing of nuclear programmes; the result has been in sharp contrast to the "too cheap to meter" claim of the 1960s. A notable exception to this trend is the French nuclear power programme which currently provides 70% of the country's electricity needs. According to Fussell:

> "One question that arises in the prevailing atmosphere of doubt about nuclear power is whether we have the technical tools for assessing and improving system reliability and plant safety in such a way that we can begin to allay public fears and restore financial credibility to nuclear power" [Fussell 1984]

Since WASH-1400 [Nureg 1975] the emergence of useful failure data concerning nuclear systems had, by 1984, brought about the publication of 22 US nuclear power plant studies [Fussell 1984]. PRA had become firmly established as a means of evaluating such risks. One conclusion emerging from these studies, along with analyses of the TMI incident, is that public risk is 50% dependent upon human performance. This dependence was confirmed by the Chernobyl disaster in 1986, where operators responding to technical personnel requests deliberately disabled safety systems. This disaster, and the contamination of the Rhine by pesticides from the Sandoz factory fire in Basle, have also illustrated the international nature of major hazards.

In a large number of modern-day disasters, for example Flixborough, TMI, Chernobyl, Challenger and Piper Alpha, human error has played a major part. To incorporate human factors in risk assessments or hazard analyses it is necessary to identify and quantify these errors and to attempt to identify their causes. In the past two decades a number of human reliability assessment techniques have been developed, the best known of which is THERP - the Technique for Human Error Rate Prediction, developed for the nuclear industry. More recent techniques attempt to incorporate expert judgement in the prediction of human error rates and causes, for example the Success Likelihood Index Method (SLIM) [Whalley 1987]. In these techniques ergonomic criteria are used to assess those environmental and human characteristics which influence personnel performance, these are known as Performance Shaping Factors [Whalley 1987].

In the process industry as well as the nuclear industry, attempts are being made to standardise methods of improving safety and reliability and of evaluating operational risks. The report of the International Study Group on Risk Analysis [ISGRA 1985] presents a formalization of the concepts used in risk analysis and suggests the use of HAZOP and FTA as the principal methods of numerically evaluating risk. In this latter respect the Chemical Process Industry (CPI) lags behind the nuclear industry although a bridging between HAZAN (CPI) and PRA (Nuclear Industry) is being seen in the UK nuclear reprocessing industry [Hensley 1985].

There is now available a wide range of computer programs for evaluating the complexities and problems of manually synthesised fault trees [Lee 1985]. Although some progress has been made towards automated synthesis, perhaps more significant to date is the appreciation of the value of the insights gained by the engineers who conduct fault propagation studies. Human understanding has also been found necessary in ensuring the correctness of the fault models produced.

In recent years, the term 'expert system' has become widely, and perhaps incorrectly, used to describe programs which are not truly knowledge based or expert. One example is the class of programs which, only a few years ago, were termed Decision Support Systems (DSS). DSSs are becoming available which are able to assist nuclear power station control personnel in fault diagnosis and mitigation [Nelson 1984]. The information utilised by these systems is primarily derived from standard operating manuals and fault diagnosis procedures. Those DSSs which can be said to be truly knowledge-based also attempt to incorporate the experiential, heuristic knowledge and procedures utilised by accomplished personnel. Considerable research has been undertaken into automated Fault Diagnosis and Alarm Analysis systems, both algorithmic and knowledge based. These systems commonly rely on incorporated fault propagation models or knowledge. Research is currently being undertaken to synthesise fault propagation information using knowledge based systems, the subject of this thesis.

If the 1970s was the decade of Fault Tree Analysis, the 1980s has so far seen reliability engineers concentrating on the evaluation of network reliability [Coppola 1984]. Communication, in terms of electrical, distribution and computer networks can be considered topologically as a collection of nodes connected by links, but, due to the exponential relationship between the number of nodes in a system and the amount of 'number crunching' required, efforts have been directed towards methods of network decomposition and algorithms for network reliability approximation [Aggarwal 1982].

This emphasis on approximation of system reliability is made not only in the face of computational complexity but also because of the inaccuracies of available primary failure data. These data can generally be expected to fall into an error band with a range of a factor of 2. This being the case, resultant system reliability can be no more accurate (few people express confidence in the accuracy of statements such as 'failure once in a million years').

A number of the major disasters that have occurred in recent years have illustrated the importance of Safety Management Systems (SMS) in the face of potential hazards. The space shuttle Challenger, which suffered catastrophic failure in 1987, is a case in point. The technology was at fault, engineers were aware of the problem, but NASA failed to take corrective action. The Bhopal disaster in 1984, causing the deaths of thousands, may have come about as a result of poor design and management on behalf of Union Carbide. No matter how refined reliability technology becomes, it is valueless unless it is implemented.

In the past there has perhaps existed the unfortunate opinion that considering safety leads only to increased expenditure (on expanded instrumentation schemes for example). It is now clear that improving reliability, up to an appropriate level, leads not only to increased safety but also to higher efficiency and improved industrial relations, with corresponding economic implications.

If a high standard of Safety and Reliability is required, this currently involves the application of techniques such as HAZOP, Fault Tree Analysis and other complex analyses. These techniques are expensive and time consuming. As the cost of computers continues to fall, the future is likely to see their increasing application as aids to the analyst in these areas. Conventional data processing approaches to the more expert areas such as Fault Tree Synthesis appear to have met with very limited success. The nature of the expert approach suggests the application of the technology of Intelligent, Knowledge Based Systems which emerged during the later 1980s. A review of this subject is presented in the next chapter.

2.3 SYSTEM SAFETY AND RELIABILITY

2.3.1 Literature Overview

The bibliography contains references which bear on the subject matter of this thesis, including those which are detailed in the text. Due to limitations of space, this bibliography should not be considered complete, although efforts have been made to ensure that pertinent references are included.

For the items in the bibliography concerned with Safety and Reliability the following series of diagrams (Figures 2.3.1b-g) present a hierarchical classification according to subject matter. This is not a general classification but focuses on the specific area of fault propagation analysis, presenting:

• Safety and Reliability	SSR	Figure 2.3.1b
• Risk Analysis	RA	Figure 2.3.1c
• Causal Analysis	CA	Figure 2.3.1d
• Fault Tree Synthesis	FTS	Figure 2.3.1e
• Fault Diagnosis	FD	Figure 2.3.1f
• Human Factors	HF	Figure 2.3.1g

The classification is outlined in Figure 2.3.1a.

Those references providing a further literature review or bibliography are underlined. There is an approximate correlation between the classification and the sub-headings of this chapter; the diagrams may be used as an overview when reading the appropriate section.

FIGURE 2.3.1a Outline of Literature Classification

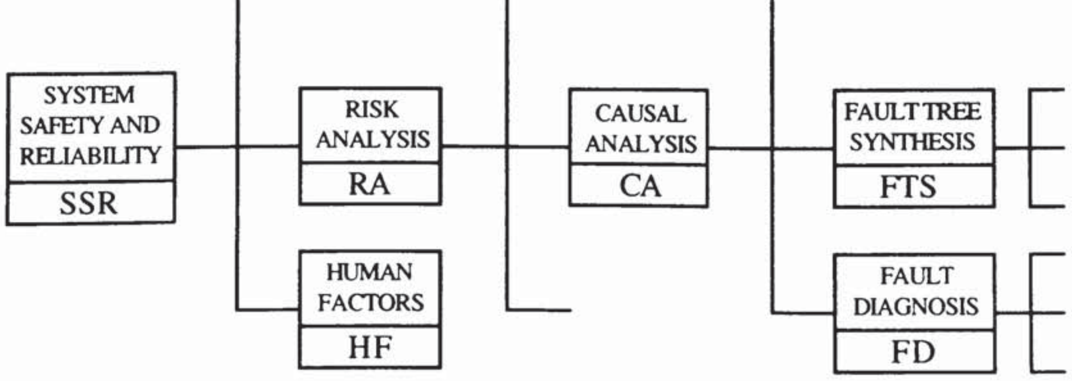


FIGURE 2.3.1b <u>System Safety and Reliability SSR</u>

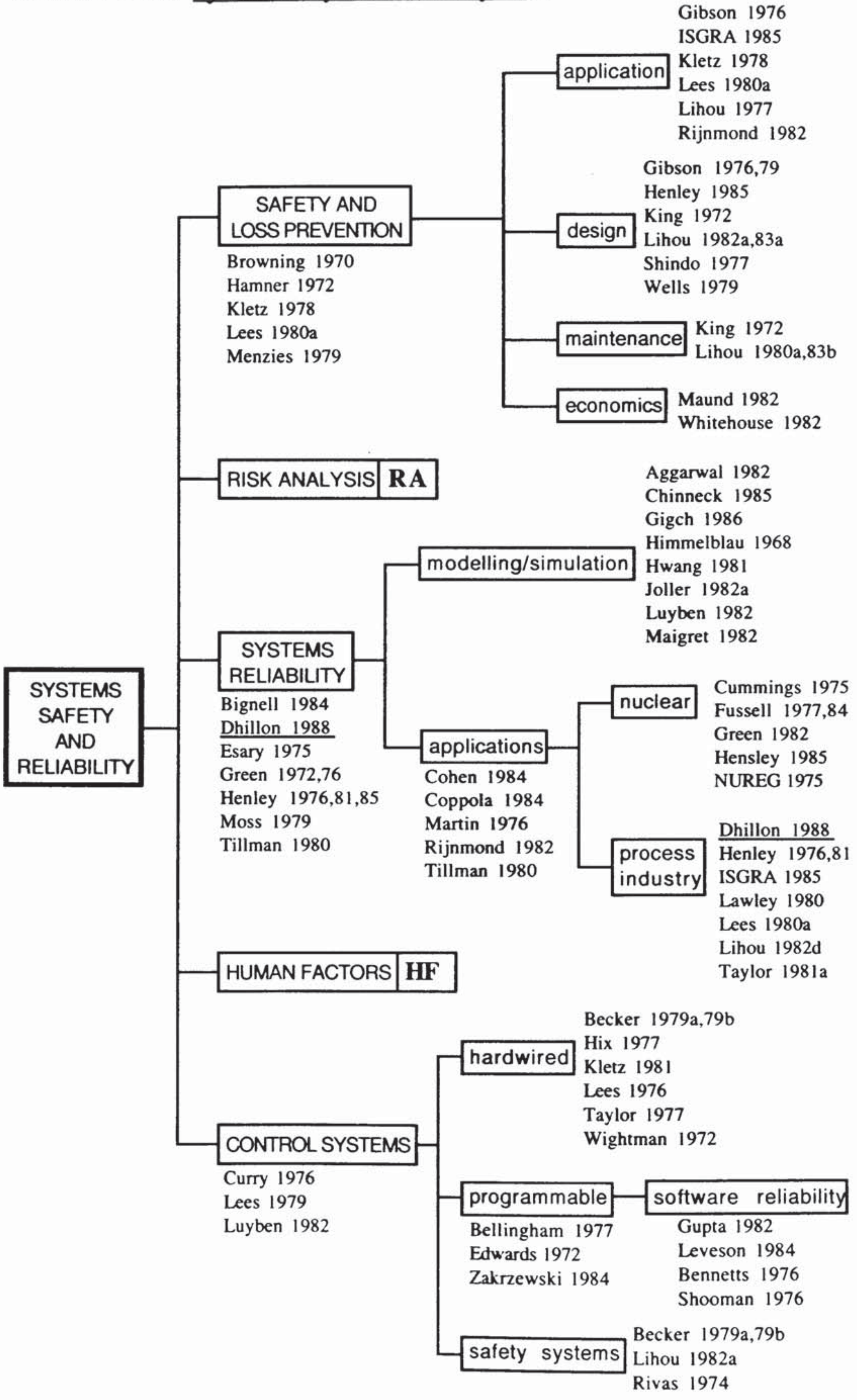


FIGURE 2.3.1c Risk Analysis - RA

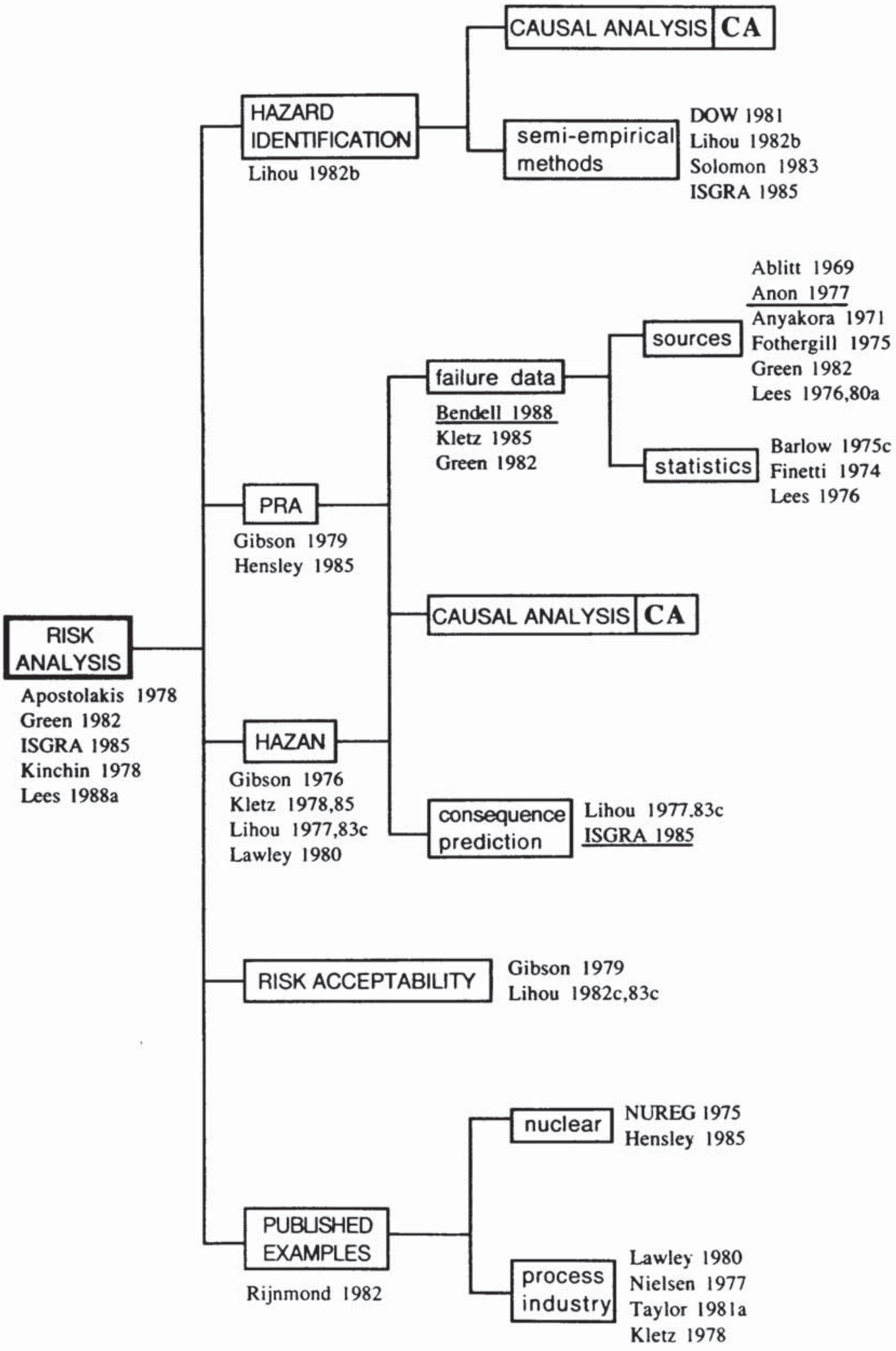


FIGURE 2.3.1d Causal Analysis - CA

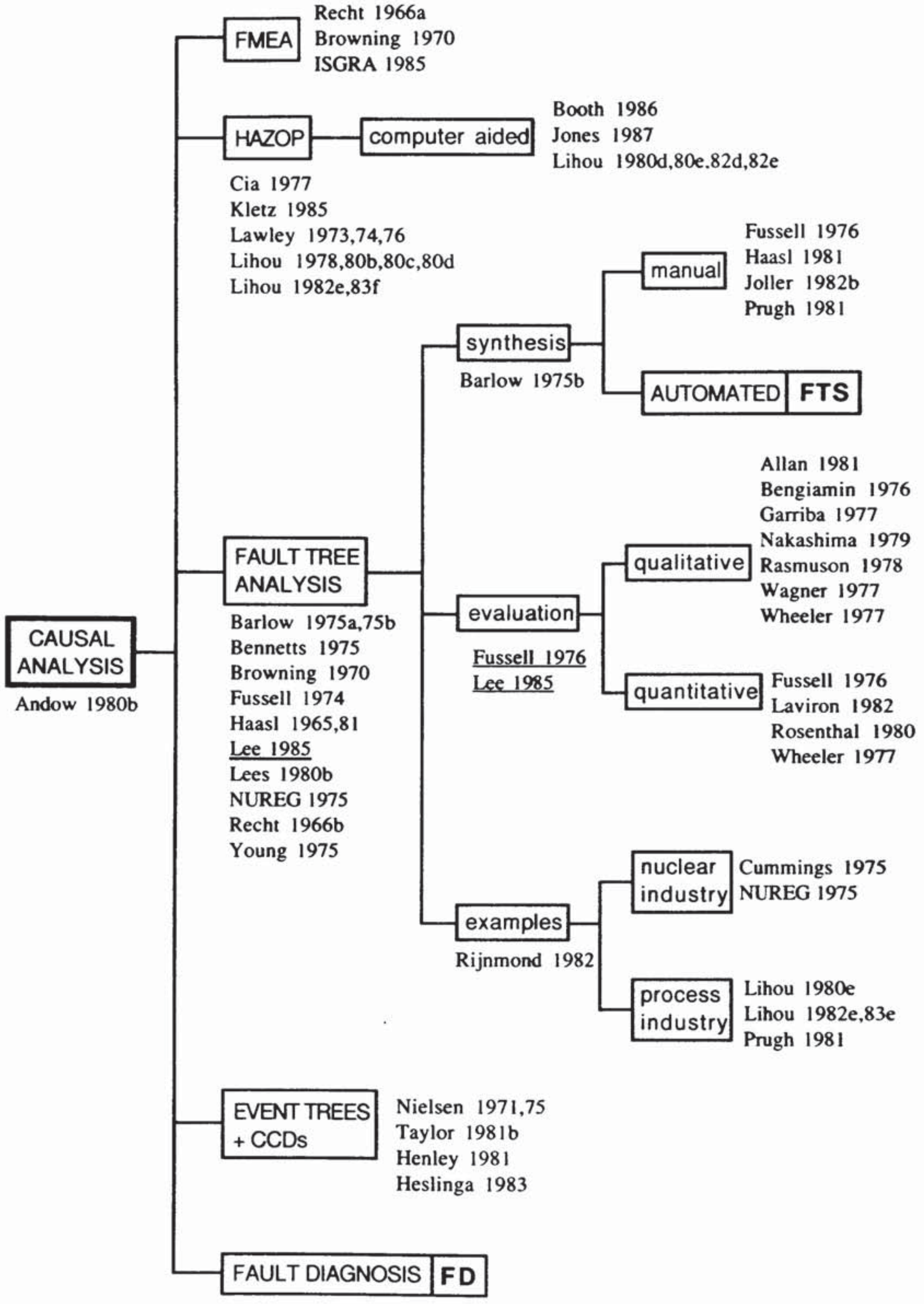


FIGURE 2.3.1e Fault Tree Synthesis - FTS

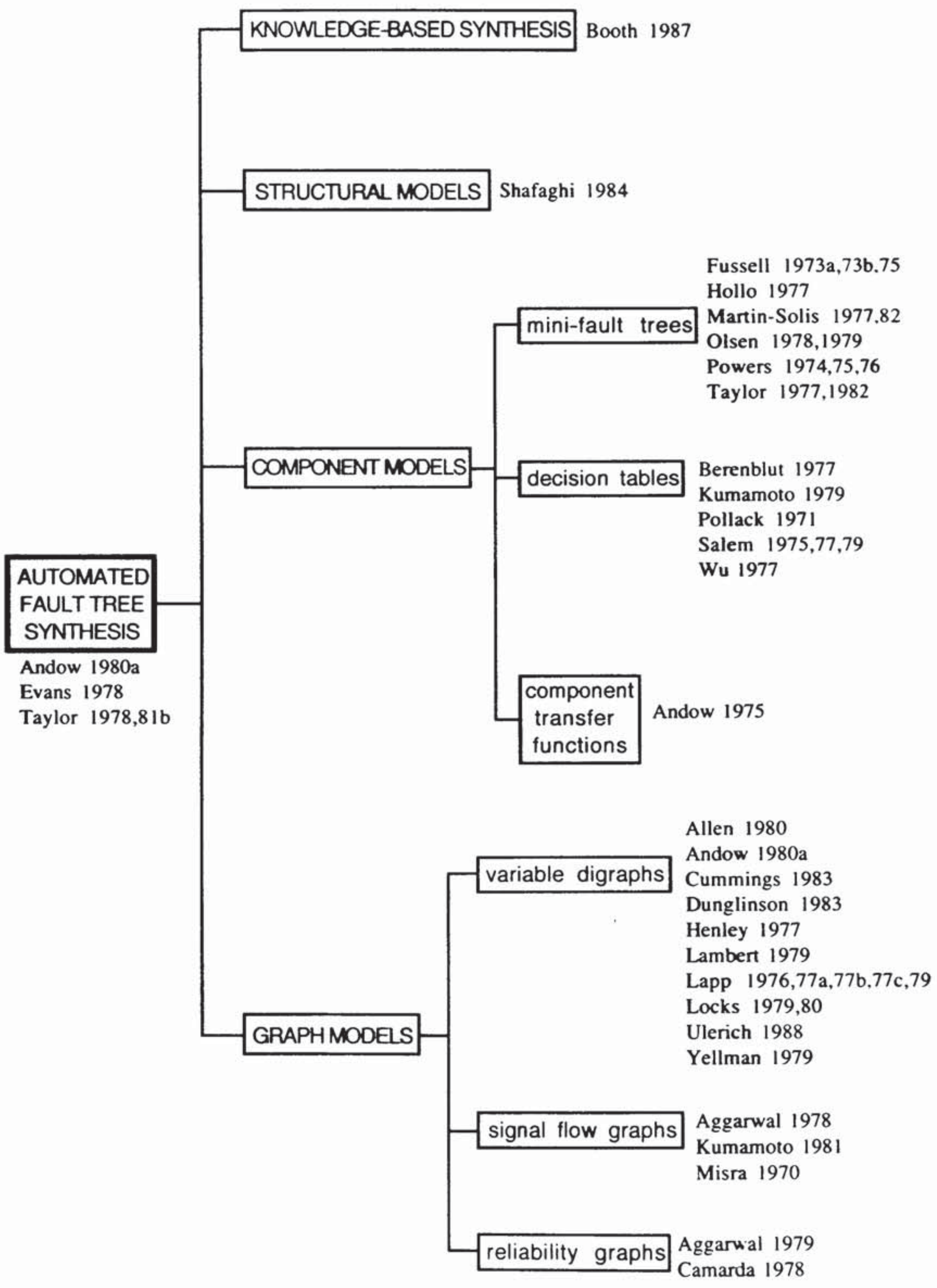


FIGURE 2.3.1f Fault Diagnosis - FD

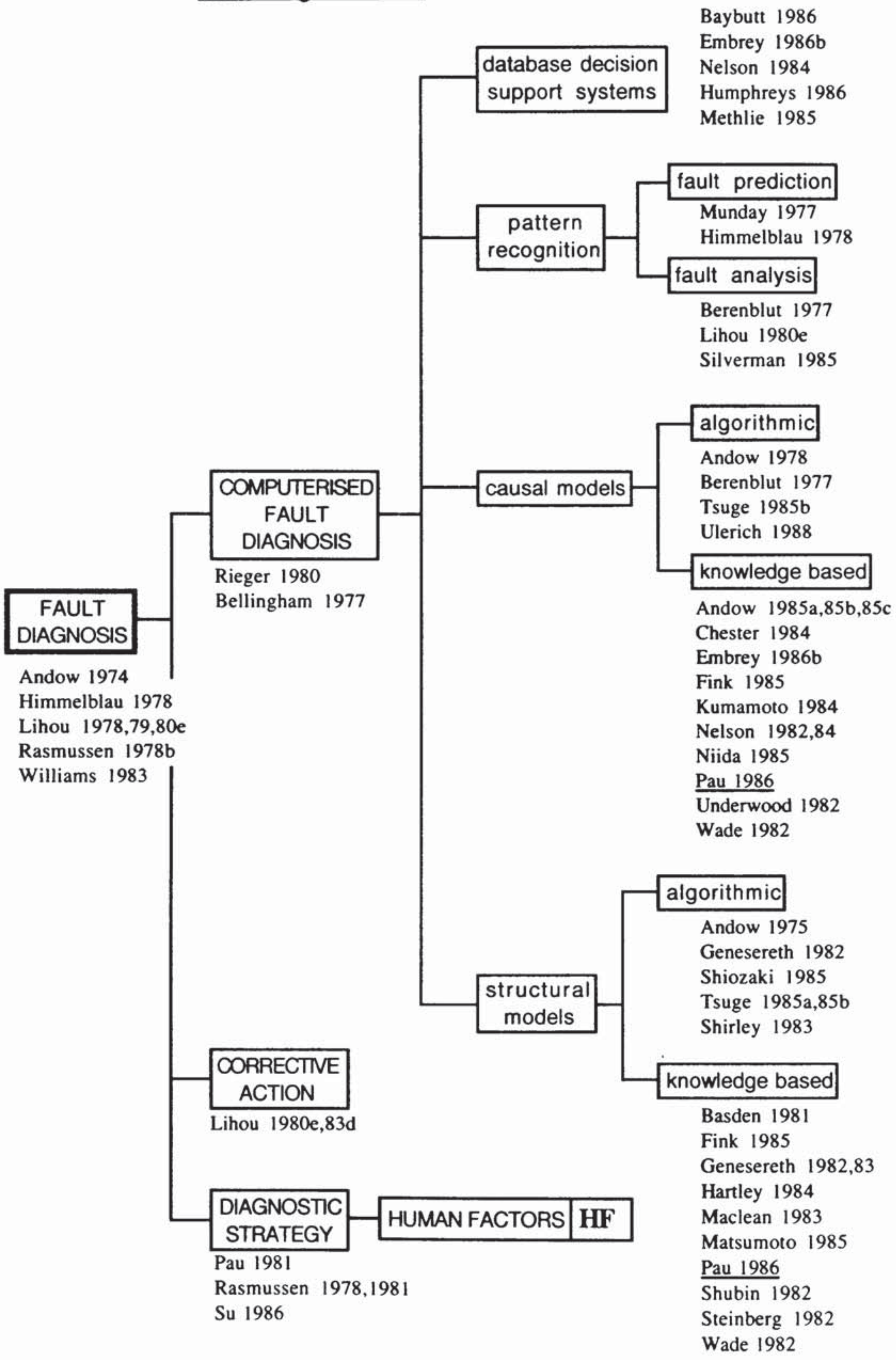
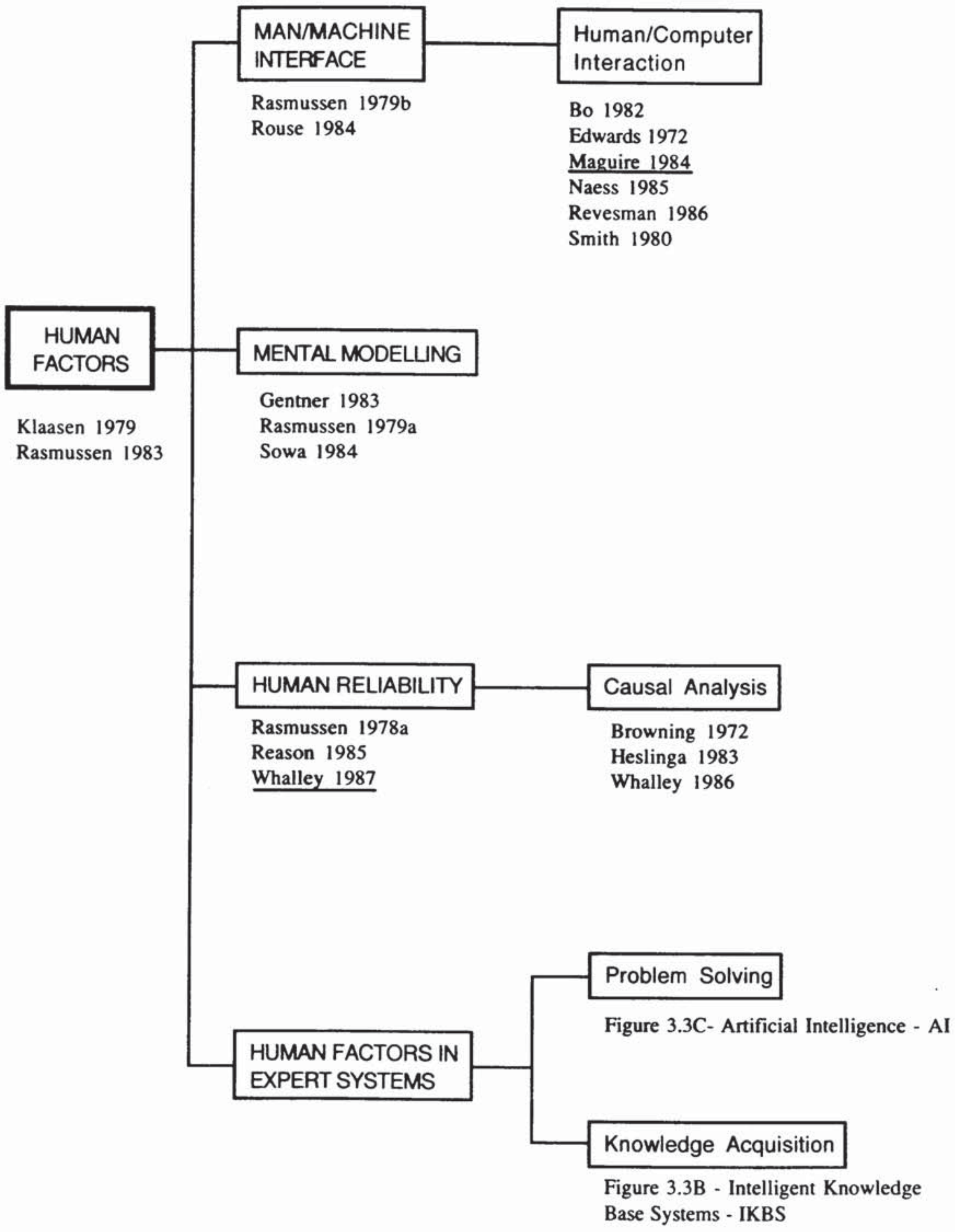


FIGURE 2.3.1g Human Factors - HF



2.3.2 **PRA / HAZAN**

Probabilistic Risk Assessment (PRA) and Hazard Analysis (HAZAN) involve the quantification of the Risks or Hazards associated with a particular activity. There are essentially four stages:

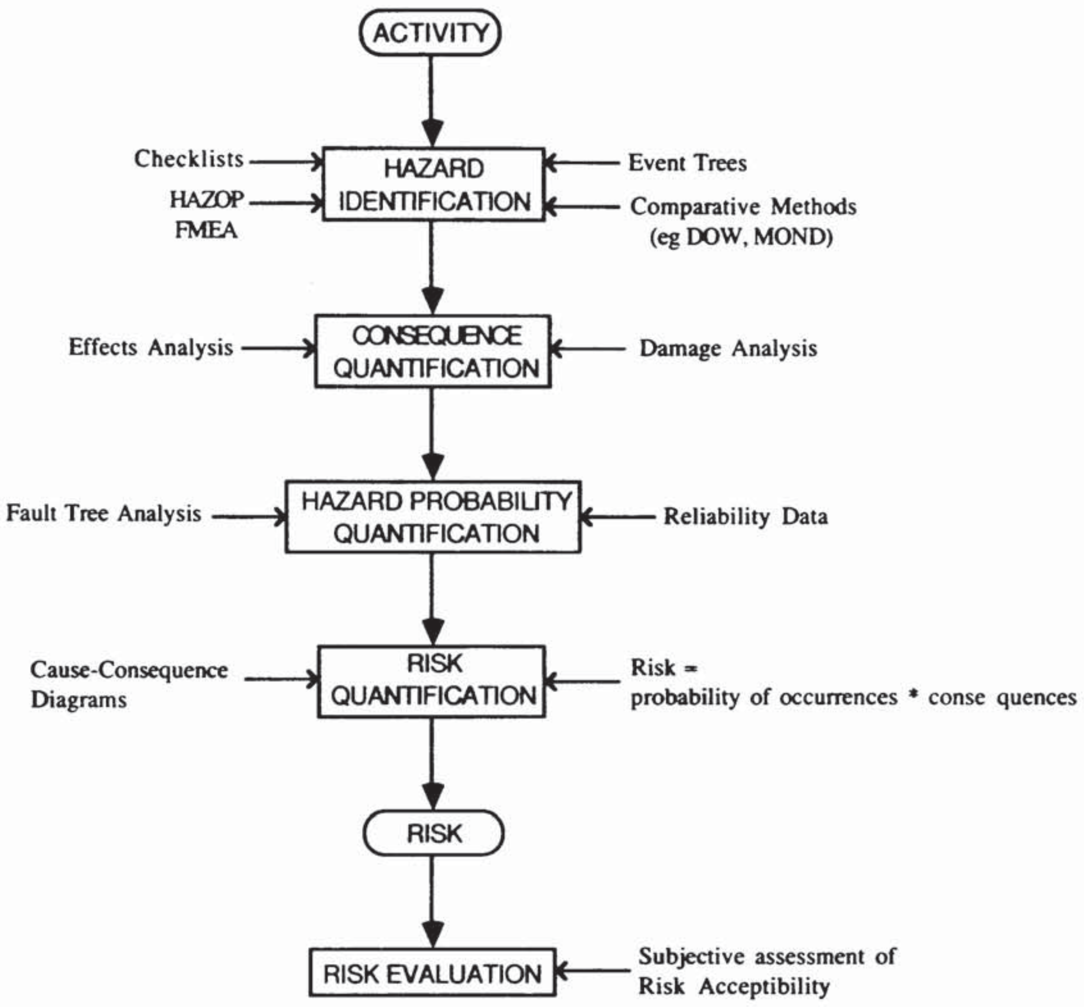
1) Hazard Identification
2) Consequence Quantification
3) Quantification of probability of occurrence of Hazards
4) Quantification of Risk

The quantification of the risk is a product of the results of stages 2 and 3:

$$\text{RISK} = \begin{bmatrix}\text{Probability of occurrence}\\ \text{of unwanted events}\end{bmatrix} * \begin{bmatrix}\text{a measure of the consequences}\\ \text{of the unwanted events}\end{bmatrix}$$

Figure 2.3.2a gives a possible sequence of the stages that may be involved in quantifying the risk associated with an activity in the chemical process industry. Figure 2.3.1c gives a hierarchical breakdown of some of the literature in this area.

FIGURE 2.3.2a Risk Assessment


ACTIVITY
Checklists
HAZOP
FMEA
HAZARD IDENTIFICATION
Event Trees
Comparative Methods (eg DOW, MOND)
Effects Analysis
CONSEQUENCE QUANTIFICATION
Damage Analysis
Fault Tree Analysis
HAZARD PROBABILITY QUANTIFICATION
Reliability Data
Cause-Consequence Diagrams
RISK QUANTIFICATION
Risk = probability of occurrences * conse quences
RISK
RISK EVALUATION
Subjective assessment of Risk Acceptibility

As may be seen from Figure 2.3.2a, Risk Assessment involves a number of techniques which analyse the propagation of faults in process plant: HAZOP, Event Tree Analysis, FMEA and Fault Tree Analysis.

These techniques involve a considerable investment in human resources. This research is concerned with the identification of the generic problems of fault analysis with a view to providing automated support for these techniques using Knowledge Based or Expert Systems technology.

2.4 HAZARD AND OPERABILITY STUDIES

This section deals with the following aspects of HAZOP: its background, the technique itself and its application, difficulties experienced with HAZOP, and more recent developments which contributed to the initiation of this research.

2.4.1 Background

By the late 1960s hazard assessments were beginning to accompany all stages from project conception through to operation of new plant; this was exemplified in the Hazard Studies of ICI [Gibson 1976, Lihou 1977]. In recognition of the fact that failure to identify certain hazards is often due to omissions in the face of complexity rather than a shortfall of knowledge or information attempts were made to devise systematic methods of identifying and evaluating process deviations. One such method, evolving from Failure Mode and Effect Analysis (FMEA- section 2.6) [Browning 1970], and ICI's own 'Critical Examination' [Kletz 1985], became known as the Hazard and Operability Study (HAZOP), which came into use in the early 1970s [Lawley 1973,74,76]. This technique has since found widespread use as the principal method of identifying hazards in the process industries.

2.4.2 Application of HAZOP

The aim of a HAZOP is to identify all major hazards and operability problems that could arise as a result of process failure or maloperation.

A HAZOP of a new design is typically carried out using a P&ID or line diagram as soon as possible after it has been completed, so that necessary modifications can be carried out before detailed design commitments are too far advanced. For existing plant undergoing a safety review it is important to ensure that the P&ID is accurate and up-to-date, reflecting modifications from the original design. HAZOP may be applied to preliminary process flowsheets early on in the life of a project; in principle the technique is applicable at all levels of detail.

A HAZOP team consists of several engineers and managers with experience or knowledge relevant to the process under investigation, along with a chairman accomplished in the technique and responsible for comprehensive recording of the results. Kletz [1985] has produced a useful guide for engineers which discusses the application and organization of HAZOP in greater detail and in the context of HAZAN (Hazard Analysis) of which it forms a part.

HAZOP studies are so named because their purpose is to identify potential hazards and operating problems. Analysis of the results from ten studies revealed the breakdown of these problems to be as follows [Lihou 1980b].

TABLE 2.4.2a Problems Revealed by Analysis of Results from Ten HAZOPs

	Timing of Study	
	Design	Operation
Unexpected operating problems	57%	33%
Hazardous malfunction	26%	40%
Operating problems causing hazards	17%	27%

These data show that operability studies tend to reveal more operating problems than potential hazards. This illustrates the key role that HAZOP can play in formulating (or evaluating) operating instructions, fault diagnosis procedures and maintenance schedules. The most common result of the utilization of HAZOP results is the reappraisal and redesign of control and safety systems, rather than the revelation of major equipment design flaws. In high risk operations, or in the evaluation of a major hazard, however, HAZOP plays only a preliminary role as a precursor to detailed quantitative causal analysis, ie HAZAN or PRA.

2.4.3 HAZOP Procedure

The first step in any HAZOP is the assembly of case-specific information relating to the plant or design under consideration. This consists principally of:

1) Comprehensive system charts or flowsheets, commonly P&IDs.

2) Expected operating specifications: physical, chemical and temporal parameter envelopes, eg materials, flowrates, temperatures, reaction data, phase data, control parameters etc.

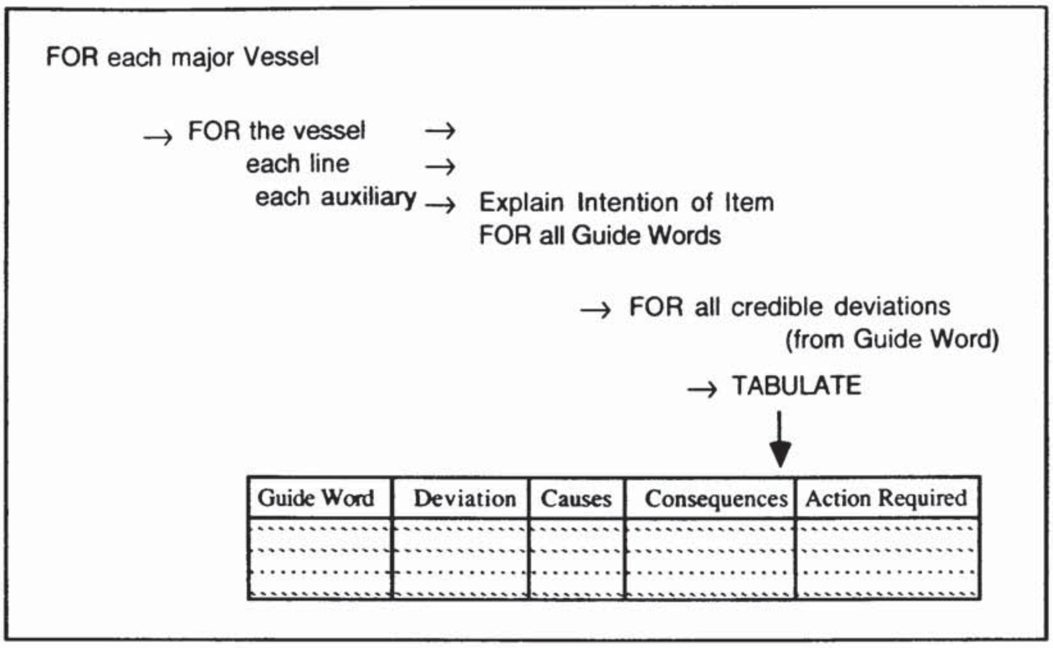
3) Case-specific equipment operating and failures modes, design envelopes and fault propagation characteristics.

A more detailed breakdown of the knowledge used in HAZOP may be found in Chapter 5 which also discusses the specific and generic knowledge brought to the study in the form of the expertise of the participants.

The basic procedure in a HAZOP is to consider all plant items - vessels, lines and auxiliaries - in a systematic fashion, commonly based on a P&ID. The study team considers all possible deviations from the intended operating conditions and looks for their causes and consequences, noting any action to be taken to mitigate revealed problems. The study is of the brainstorming type in order to prevent preconcieved or overly structured thought processes leading to significant omissions. As a result of workload pressures, however, it is common practice to record only those events which (in the eyes of the analysts) could lead to a potential hazard or major operability problem.

The detailed sequence of an operability study with the systematic breakdown of the P+ID into individual items is illustrated in the flowsheet of Figure 2.4.3a. For each vessel, all lines and auxiliaries are studied in sequence, usually following the direction of flow of the main process stream. As a preliminary to the consideration of any item, its function and/or purpose is noted, along with key operating conditions; this focuses attention on deviations which contravene these design intentions and ensures that quantitative requirements and limitations are borne in mind.

FIGURE 2.4.3a Detailed Sequence of a HAZOP Study



To facilitate consideration of all deviations a set of guide words, representing qualitative change (eg more, less), are applied to each plant item. A list of seven guide words recommended in the CIA guide to HAZOP [CIA 1977] are explained in Table 2.4.3a. Each guide word is considered in turn, and is applied to process variables or operations pertinent to the particular item to produce qualitative deviations from normal conditions (eg more flow). If a given deviation (event) is considered both credible and significant, its causes, consequences and indicators are noted, along with an evaluation of hazards or operating problems and actions determining appropriate prevention or limitation of these problems. The results are tabulated under headings such as Guide Word, Deviation, Causes, Consequences and Action Required. An early example of a HAZOP study was published by Lawley [1974].

TABLE 2.4.3a <u>HAZOP Guide Words</u>

Guide Word	Meaning	Comments
NO or NOT	The complete negation of the intentions	No part of the intentions is achieved; nothing else happens
MORE LESS	Quantitative increases or decreases	Applicable to quantities and properties such as flow and temperature as well as activities such as 'heat' and 'react'
AS WELL AS	A qualitative increase	All intentions are achieved together with some additional activity
PART OF	A qualitative decrease	Only some of the intentions are achieved; some are not
REVERSE	The logical opposite of the intention	Mostly applied to activities, eg reverse flow, chemical reaction or heat transfer
OTHER THAN	Complete substitution	No part of the original intention is achieved. Something quite different happens

2.4.4 Difficulties With HAZOP

A common objection to HAZOP is that it may result in expensive plant or design modifications. This viewpoint is easily refuted: "the main objection to visiting the doctor is that it may result in expensive bills for treatment" [Kletz 1985]. There are, however, a number of genuine problems associated with HAZOPs conducted in the fashion described above:

1) Although guide words provide for the inclusion of all qualitative aspects of change (eg more, less) there is no corresponding checklist to cover all of the process conditions and operations that are subject to change (eg flow, pressure). Similarly there is no systematic formalization of equipment failure modes.

2) Considerable effort (and therefore expense) is required to produce comprehensive HAZOP results. Shortcuts or omissions reduce the confidence that all major losses have been included.

3) Whilst discussion of minor event sequences may take place, only "important" event sequences are usually recorded.

4) For each entry in a HAZOP table there is a tendency to directly relate each event back to its root causes (primary failures), or forward to its ultimate consequences (major losses). This means that the details of fault propagation are not recorded.

5) Each entry in a table represents a discrete event sequence. In conjunction with 3 and 4 above, this produces difficulties in uncovering remote interactions; the team may be forced to flit backwards and forwards over the P&ID in order to identify the interactions. In cases where intervening causal chains are omitted due to their lack of "importance" the team cannot use previously noted records in tracing an event to a remote cause. Significant omissions may occur as a result of these problems.

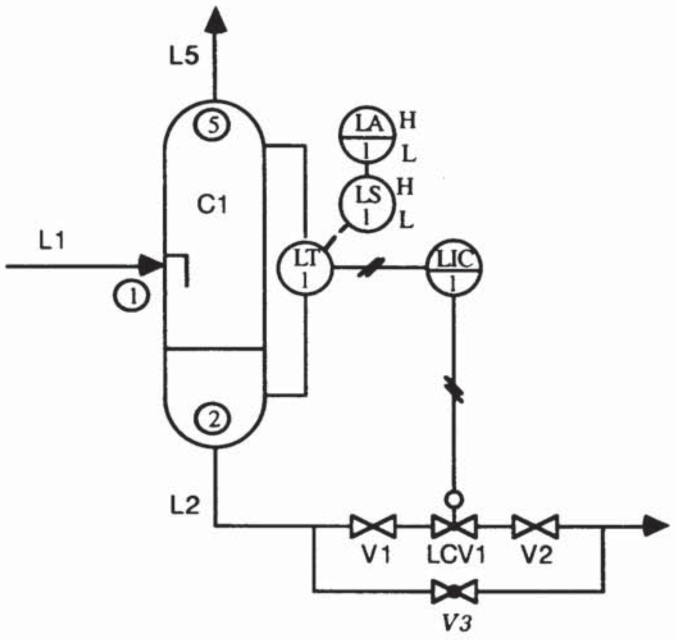
6) Any modification of the original design necessitates a re-evaluation of the HAZOP. It is often impractical to completely redo a section of plant and a piecemeal approach may lead to significant omissions, especially in the case of remote interactions.

7) A problem common to many types of qualitative analysis:- each analyst or team will almost certainly produce a different result. Because of the lack of a formal causal structure, these results are difficult to compare and to verify. This exacerbates problem 6 as the original HAZOP team is usually not available.

2.4.5 Recent Developments

In the late 1970s, workers at Aston University tackled these problems. The technique devised by David Lihou [Lihou 1977,78,80c, Jones 1987] is described. This is illustrated by an example: the Ammonia Let-Down System (ALDS), part of which is shown in Figure 2.4.5a, drawn from reference [Lihou 1982d] which is reproduced in full in Appendix 1.

FIGURE 2.4.5a Ammonia Let-Down System - Two Phase Splitter



Progress was made in three stages: formalization of events and failure modes, recording causal information in the form of logical equations, and the linking together of the equations using computer aids to produce fault trees for further analysis.

1) Formalization of Events and Failure Modes

An **event** is defined as follows:

event = **object** (**property** word , **guide** word , {**component**})

e.g. Line 1 (flow , more , syngas)

Object refers to the physical location of the event (vessel, line, valve etc). Guide Words (no, more, less etc) are as defined in Table 2.4.3a, and are applied to Property Words: the process variables and conditions (flow, temperature etc) shown in Table 2.4.5a. The names of process materials (components)are included where appropriate. Thus an example of an event is "Line 1 (flow, more, syngas)", as above. For rapid recording of the results, property words, guide words and components may be referred to by index numbers. The complete set of index numbers for the ALDS is shown in Table 2.4.5b. The example event becomes "L1(133)". In considering a particular plant item, the team may prefer to consider each property in turn, applying the set of guide words to it. In this case, the first two columns in the tabulated records would read property word, guide word, rather than guide word, deviation.

TABLE 2.4.5a Property Words

Process Conditions	Process Operations	Physical Properties
Flow	Separate	Density
Temperature	Heat Transfer	Viscosity
Pressure	Mix	Vapour Pressure
Level	Absorb	Calorific Value
Concentration	React	pH

TABLE 2.4.5b Index Definitions for Events in ALDS

Index	Property Word	Guide Word	Component
1	Flow	No	Ammonia Liquid
2	Pressure	Less	Ammonia Gas
3	Temperature	More	Synthesis Gas
4	Level	As well as	Methane
5	Concentration	Part of / fluctuation	Instrument Air
6	Separate	Reverse	
7	Heat Transfer	Other than	

Relevant equipment failure modes are also tabulated, and may be referred to by code numbers or letters. Those for the ALDS are shown in Appendix 1.

The importance of these tabulations is that, in principle, all plant fault states (process deviations and equipment faults) may be defined *a priori*. This provides a structure for the inclusion of all such faults, ensuring that none are omitted.This complete set is, of course, limited by the context of the plant item(s) under consideration at any particular time.

2) <u>Cause and Symptom Equations</u> (CSET)

As an alternative to tabular records Lihou [1980c,d] suggested that causal links could be recorded in the form of logical equations:

Cause equations:-	in which a process deviation is equated to the logical combination of its causes,
Symptom equations:-	where deviations at the inputs to process vessels are equated to their consequential effects in, or at the outputs of, the vessel

In order to explicate causality within and through vessels, appropriate points or streams within vessels are defined as Nodes, serving as locations for the definition of events thereby facilitating symptom equation formulation.

In the ALDS P&ID, Figure 2.4.5a, we may consider the causes of no flow in line 2:

Line 2 (flow,no) **caused by** : Node 2 (flow no)
or Line 2 (blocked)
or Line 2 (blocked valves)

This cause equation has the coded form: L2(11) = N2(11) + L2(0) + L2(BV)

Looking at the consequences of no flow into C1:

Node 1(flow,no) **causes** : Node 2(flow,no)
and Node 5(flow,no)

which may be represented by the symptom equation: N1(11) - N2(11) * N5(11)

The operators used here (=,-,*,+) distinguish cause and symptom equations and denote 'and' and 'or' logic relations. The full set of equations for the ALDS is reproduced in Appendix 1.

The use of cause and symptom equations has a number of advantages:

i) Provision is made for consideration of all deviations and failure modes.

ii) Remote interactions may be thoroughly explored - fault trees may be constructed directly (see below).

iii) The effects of plant modifications may be quickly assessed by rewriting the appropriate equations.

iv) There is a considerable saving in the amount of time (and therefore money) required for the study.

v) At any one time the team's attention is focused on a limited section of plant. This allows for concentration on a section with clearly defined boundaries, rather than flitting from place to place on the P&ID exploring remote interactions. Remote interactions are uncovered by equation links.

vi) Using a computer gives a permanent but modifiable readily accessible record.

vii) Computer oriented structured synthesis raises the possibility of automated synthesis.

3) <u>Fault Trees from Operability Studies</u>

In a comprehensive HAZOP study using cause and symptom equations, all significant events and causal interactions should be described. It is then possible to produce a fault tree directly for a desired event by tracing the causal logic through the equations. Obviously, this is much easier to do by computer than by hand, and suitable software has been written. In the CAFOS program [Jones 1987] the equations are interpreted into a suitable form for fault tree analysis. The fault tree may be displayed pictorially for qualitative assessment, and probability calculations may be carried out based on available primary failure data. The author has subsequently written a personal computer version of the program - PC-CAFOS [Booth 1986]. Both packages have found commercial applications.

Detailed discussion of fault tree analysis and fault tree synthesis may be found in section 2.5. It may not have escaped the reader's attention, though, that the technique described above is significantly different from conventional HAZOP. Indeed it may be said to be closer to fault tree synthesis, as the end result is explication of all events and causal relations rather than just those of significant interest. Furthermore, generation of cause and symptom equations is not accompanied by an immediate analysis of revealed hazard states; there is no 'action required'.

In a fault tree synthesis (section 2.5.3) a fault tree is developed for a single identified undesirable event. In CSET (in principle) all fault trees are generated in one go. As the equations are generated by the HAZOP team they are able to identify those undesirable events which are worthy of further consideration. Once the set of equations is complete, these events may be reappraised in the light of the full fault trees that are now available. The computer aid allows exploration of remote interactions which may well have been missed in a conventional HAZOP. For those events which are selected for detailed HAZAN, fault trees are already available for qualitative and probabilistic evaluation.

It must be stressed, however, that confidence can only be placed in the quality of the fault trees if the completeness and logical correctness of the CSET study is assured. If there are any omissions in the equations, events will fail to "link up" in the fault tree. Any malformed equations will result in illogical fault trees. In order to address these problems Lihou [1982e] has used general rules to assist in cause equation formulation; an example is given in Figure 2.4.5b.

FIGURE 2.4.5b An Example of a Cause and Symptom Equation Rule

Pipelines - Flow

No flow may be caused by any of the following:-

- No flow in the line(s) immediately upstream
- No flow at the node where the line leaves an equipment
- The supply tank empty
- A valve shut in the line
- A filter fully blocked in the line
- A pump in the line stopped

It may be seen, then, that the use of cause and symptom equations provides a bridge between the techniques of Fault Tree Synthesis/Analysis and HAZOP. The systematic, formalised and structured method, the proof of ready computerisation by CAFOS and the existence of synthesis rules as above all suggest the possibility of a rule-based system ("expert" system) to aid in the analysis. This possibility provided the initial stimulus for the present research.

2.5 FAULT TREE ANALYSIS

This section deals with the following aspects of Fault Tree Analysis:

1) Background - The historical development of the technique
2) Fault Trees - An introduction to their construction and utilization
3) Fault Tree Synthesis - An in-depth discussion of both manual and automated synthesis
4) Fault Tree Evaluation - How the trees are used after construction

The term "Fault Tree Analysis" is used to describe the entire process of analysing a system using fault trees, incorporating the two stages of synthesis and evaluation. An overview of fault tree analysis literature is given in Figures 2.3.1d and 2.3.1e.

2.5.1 Background

Fault Tree Analysis was first devised in 1961 to evaluate the reliability of the launch control system of the Minuteman missile. After a safety symposium in 1965, at which a number of papers on the subject were presented [Haasl 1965], the technique was more widely investigated by reliability and safety analysts who at that time were in need of such a means of evaluating the causes of identified system failures.

In the early 1970s fault tree analysis was adopted for widespread use in the aerospace, electronic,and nuclear industries; a study evaluating US light-water reactor accident sequences - WASH-1400 [NUREG 1975] devoted 1300 pages to the subject. In proving the usefulness of the technique this also illustrated the considerable effort involved in the synthesis of fault trees - in this case, measured in man-years. A number of workers tackled the problem during the decade, meeting with some success in the development of computer-based synthesis aids (section 2.5.4).

Considerable efforts were also directed towards the automated evaluation of fault trees, both qualitative, in terms of cut-set and path-set (tie set) derivation, and quantitative - probabilistic assessment of the likelihood of faults based on available primary failure data, and statistical analyses comparing event sequences. A large number of commercially available codes were written during this period [Lee 1985]; such codes were proven to be a necessity rather than a luxury by WASH-1400.

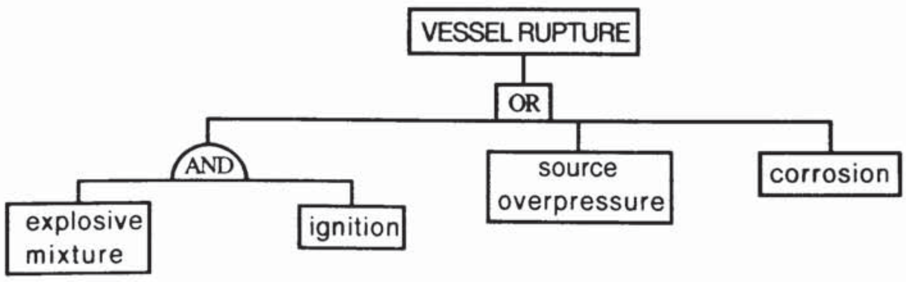
A recent review [Lee 1985] illustrates the domination of the reliability literature of the 1970s by fault tree analysis. It has emerged as the principal method of fault propagation analysis in Probabilistic Risk Assessment (PRA) and Hazard Analysis (HAZAN). Although computerised evaluation is well advanced, automated synthesis has met with limited success and has not gained widespread acceptance.

2.5.2 Fault Tree Analysis

Hazard identification procedures such as HAZOP, when used in a Hazard Analysis (HAZAN) reveal the possibility of occurrence of a number of unwanted happenings, termed events. In order to discover how these events may come about, and how likely they are to occur, it is necessary to make explicit all relevant event sequences, from primary equipment failure or human error through to the specific identified event. Fault Tree Analysis provides a means of describing and evaluating this fault propagation.

The starting point in the construction of a fault tree is the identified unwanted event, termed the "top event". The first step is to deduce the immediate causes of the top event, grouping them according to how they combine to cause it. This is illustrated in the simple fault tree shown in Figure 2.5.2a, where the causal logic is shown by AND and OR logic gates.

FIGURE 2.5.2a A Simple Fault Tree



Each leaf ("bottom event") in the tree is considered in turn, the combination of events causing it being described. This iterative process continues the downward development of the tree until "primary events" are reached. These are events such as component failures which are considered to have no causes (more specifically, they have internal failure causes which are not developed), or events which need no further resolution; failure data is available or may be estimated, or failures whose causal sequences are difficult to identify (many human errors fall into this category); failure rates must be estimated from the best available experience. Events whose causes lie outside the defined system (usually occurring at the physical boundaries of the system) fall into this category.

Once constructed, a fault tree may be evaluated qualitatively - by inspection or by derivation of event cut sets and path sets, and quantitatively - primary event failure data may be used to directly calculate the probability of occurrence of the top event. Fault tree evaluation with respect to the verification of the synthesis is discussed in section 2.5.5.

Fault tree analysis is of value in:

- Helping the analyst to understand the system and the implications of the chosen design
- Assessing hazard or risk levels.
- Assessing system availability / mission success rate.
- Directing safety/reliability resources to areas where they are best employed.
- Comparing different designs.
- Identifying and analysing common mode failures.
- Identifying events critical to system reliability and thus components critical to system operation.
- Compiling operating manuals, fault diagnosis aids and maintenance schedules
- Demonstrating compliance with safety requirements.
- Justifying design changes.
- Evaluating the effectiveness of safety systems.

The use of fault trees is discussed in [Young 1975].

2.5.3 Fault Tree Synthesis

Fault tree synthesis encompasses a number of procedures:

1) System Definition

Before constructing a fault tree the analyst must possess a thorough understanding of the system under consideration; all relevant information must be available:

- comprehensive system charts or flowsheets: eg circuit diagrams or P&IDs.
- process specifications (materials, design and operating parameters etc)
- component operating and failure modes and fault propagation characteristics.
- system boundary conditions (spatial and temporal) - those conditions which must or must not be present and specific operational parameters pertinent to the top event, serving as limitations on the events allowed in the context of the top event.

It is important that the top event is carefully selected and defined; the assumptions in its definition determine the system boundary conditions and thus the nature of the final fault tree. The information required here is similar to that required for HAZOP (section 2.4); Powers [1974,75] presents a classification which is of value in ensuring its completeness. A more detailed study of this knowledge appears in Chapter 5.

2) Event Definition

An **event** may be either a **fault** or a **failure**. Failures are those events which are considered primary (having no further causes - see above) whereas the causes of a fault may be developed further - these may be other faults or failures. An event is defined by its location - the item where it occurs, and the state which the item enters into. Table 2.5.3a shows the conventional event symbols as used in WASH-1400 [NUREG 1975], although there are variations on this theme [Fussell 1976, Haasl 1981].

3) Logic Gates

Although it is only necessary to use AND and OR gates in constructing a tree it is sometimes useful to employ others, shown in Figure 2.5.3b. XOR and NOT gates must be used with caution as there are problems in ensuring the logical correctness of the resulting tree. In a fault tree, all primary events should be independent of each other. Any interdependence is called s-incoherence and requires special means of evaluation [Fussell 1976, Rosenthal 1980, Joller 1982a]. It is possible (and simpler) to construct fault trees using AND and OR gates only; XOR and NOT gates should not occur in logically formed fault trees and INHIBIT gates may be replaced by AND gates without affecting tree evaluation; this simplification is generally made in automated synthesis.

TABLE 2.5.3a <u>Fault Tree Event Symbols</u>

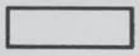
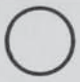
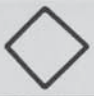
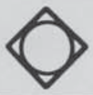
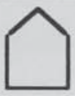
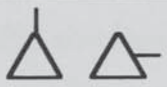
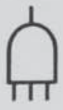
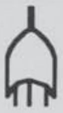
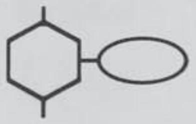
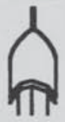
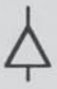
		
Fault Results from the combinations of events through the input logic gate.	Primary Failure A basic event or component failure requiring no further development.	Primary Undeveloped Considered a basic event because it is of insufficient consequence or the neccessary information is unavailable.
		
Subtree Exists A subtree exists and has been evaluated elsewhere; the quantitative results are inserted as though it were a component.	Switch/normal state A switch to include or eliminate parts of the fault tree which may or may not apply to certain situations. Also used to describe high-probability normal events.	Transfer in/out Used to modularise the tree. Applicable in the case of repeated events in the fault tree.

TABLE 2.5.3b <u>Fault Tree Logic Gate Symbols</u>

 AND	 OR	 INHIBIT	 XOR (exclusive or)	 NOT
The output event occurs if all the input events occur	The output event occurs if one or more of the input events occur	The input event directly produces the output event if the indicated condition is satisfied	The output event occurs if one and only one of the input events occurs	The output event occurs if the input event does not occur

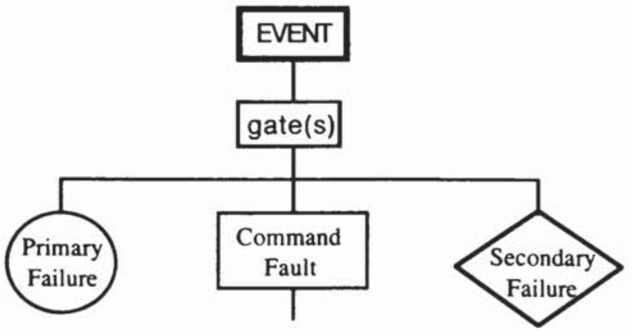
4) Tree Synthesis

Published information dealing with generalised fault tree construction (apart from automated synthesis (2.5.4)) is limited. Reliance is placed largely on the knowledge and understanding, intuition, deductive capability, thoroughness and previous experience of the analyst.

Haasl [1965,81] devised a "structuring process" to assist in the selection and application of logic gates when developing the tree. Fault propagation is developed from the functional or structural description of the system through system, subsystem and component levels. In this methodology three types of causal event are defined, as in Figure 2.5.3a:

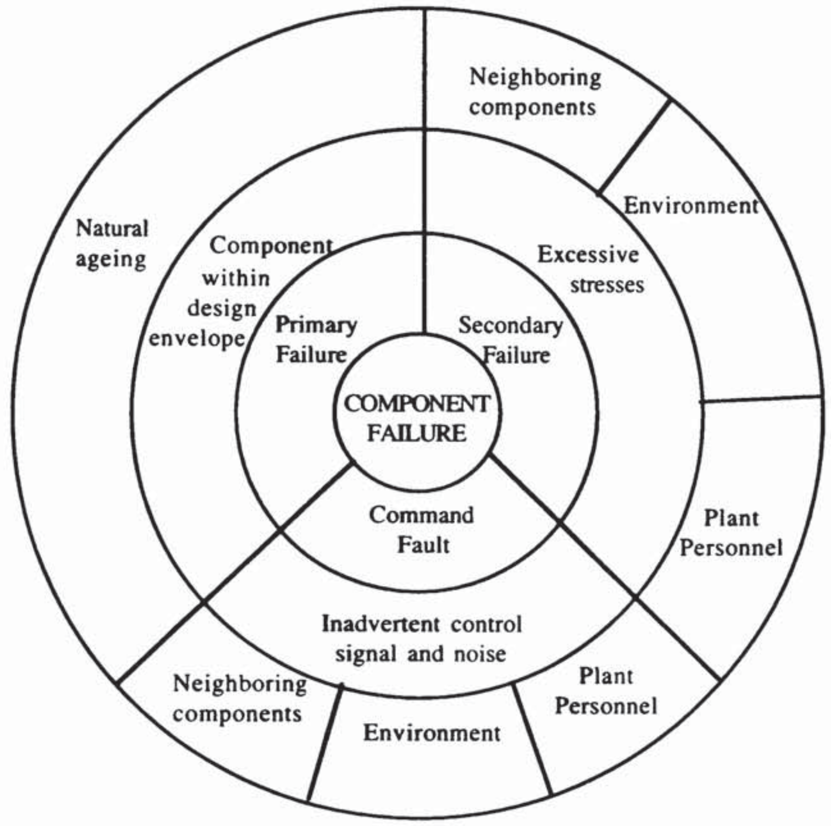
1. Primary Failure - due to internal failure of the component; when operating in an environment for which the component is qualified; causes are not further developed.
2. Secondary Failure - due to factors outside the component design envelope
3. Command Fault - maloperation of a component due to the action of an initiating element via a control signal or external normal causal connection, distinguished by the fact that the component can operate correctly if the initiating element returns to an unfailed state.

FIGURE 2.5.3a Causal Event Classification



Applying this "gate template" in conjunction with the component failure characteristic diagram Figure 2.5.3b [Henley 1981] assists in developing the causes of a particular event.

FIGURE 2.5.3b Component Failure Characteristics



From the limited publications dealing directly with manual FTS techniques [Barlow 1975b, Dunglinson 1983, Fussell 1976, Haasl 1965,81, Prugh 1981] it is possible to identify a number of guidelines, principally based on Haasl's "structuring process":

1) Ensure that the 'where', 'what' and 'when' of all events is described fully.

2) Causality should not pass through an OR gate. Inputs to an OR gate are identical to the output but are more specifically defined as to cause. If causality appears to pass through an OR gate this is an indication of a missing AND. This rule (which is largely ignored) is a reflection of the fact that a given event in a fault tree actually represents a system state , ie for that event to occur the system is in a certain fault state defined by a sequence of events occuring below the given event in the fault tree.

3) When refining an event ask: can this event consist of a component failure?

   Yes - state-of-component fault - add an OR gate and find primary, secondary, and command fault modes.

   No - state-of-system fault - look for immediate,necessary and sufficient causes through AND and OR logic.

4) A given event may be defined in terms of a failure mechanism, mode or effect, ie for a given subsystem:

   Failure Mechanism - the sequence of events internal to the subsystem which causes it to fail.

   Failure Mode - a failed state of the subsystem, defined at the subsystem level.

   Failure Effect - the effect of the subsystem failure on the system of which it forms a part.

   Refinement of an event thus passes from failure effect, through mode, to mechanism, ie the effect of a subsystem failure becomes a mechanism of a failure mode of the system.

5) No Miracles: assume "all other things normal", ie a component will always propagate a fault.

6) Complete the gate: complete all inputs to a gate before further refinement.

7) No gate-to-gate: ensure that the output of a gate leads to a defined event and not directly to another gate.

8) Repairability of components is a consideration only when evaluating the fault tree; when constructing the tree consider all components as non-repairable.

9) It is useful to distinguish between initiating and enabling events [Dunglinson 1983]. Some events initiate fault propagation whilst others permit the propagation to continue by enabling AND gates (eg a control system device failure or redundancy loss). The enabling event may remain undiscovered for some time, before an initiating event combines to cause a fault.

The top event of a fault tree is often general in nature (eg "chlorine release") and must first of all be defined in terms of system faults. Definition of the "tree top" largely determines the structure of the tree thereafter. As a number of top events such as vessel overpressure, explosion, fire etc are encountered frequently it is useful to generalise the corresponding tree tops to form templates for use in further analyses. The well known fire triangle is a simple example of such a template. Prugh [1981] has described a number of tree top templates applicable to process plant.

2.5.4 Automated Fault Tree Synthesis

2.5.4.1 Introduction

With the increasing interest in Fault Tree Analysis (FTA) in the early 1970s, it was JB Fussell [Fussell 1973a,b] who first published a formalization suitable for automated fault tree synthesis (FTS), for electrical systems. During the decade a number of workers attempted to devise such methods for the nuclear and chemical process industries leading to a number of publications which are classified in the literature overview (Figure 2.3.1e) according to the basis of the synthesis. This section presents an outline, an analysis and a discussion of FTS methods.

2.5.4.2 Outline of Synthesis Methods

As can be seen from the literature overview synthesis methods fall into two main groups, based on component models or graph models.

1) Component Models.

The system is decomposed into components or basic devices. For each component causal logic is defined - as mini fault trees, decision tables, transition tables or transfer functions; these causal models are usually stored in a generic library. Tree synthesis is achieved, essentially, by "cobbling together" mini-fault trees for individual components according to the system topology to form the overall fault tree. This assembly may be achieved with respect to constraints defined by the domain of the top event and of the events added as the tree is developed, along with rules to structure the process according to defined subsystems, event classifications, control systems and generalised logic.

2) Graph Models.

These methods are based on the topological representation of system variable and component interactions in the form of variable digraphs, signal flow graphs or the reliability graph of the system. The graph model is first synthesised from the structural model (system description, eg P&ID or circuit diagram) and then an algorithm is used to traverse the graph generating the fault tree. Generation is in the context of constraints collected in the process along with generalised control loop and structural rules. This is a process analogous to the use of constraint propagation for theorem proving in Artificial Intelligence [Charniak 1980, Sussman 1980]

3) Structural Models

More recent efforts towards component model synthesis have focused on the structuring of process systems by control loops. The plant is considered as a set of interconnected loops, whose individual fault structures are combined to give the fault tree.

4) Knowledge Based Synthesis

The research presented in this thesis is concerned with knowledge-based fault tree synthesis in respect of the similarity that exists between FTS and the method of HAZOP recording using boolean equations [Jones 1987] discussed in 2.4.4. To date, the author is not aware of any successful accounts of attempts to specifically generate fault trees using KBS. KBS are, however, finding increasing use in the development of fault diagnosis and decision support aids; relevant here are those knowledge based fault diagnosis systems which generate causal models from structural descriptions. This topic is looked into in the section on Fault Diagnosis, 2.7.

2.5.4.3 Analysis of Synthesis Methods

1) COMPONENT MODELS

1a) Mini-Fault Trees

Fussell [1973a,1973b,1975]

Fussell formalised concepts involved in the causal analysis of simple electrical systems to produce the Synthetic Tree Model - STM. This structured fault tree synthesis method, whilst providing a manual aid, was suitable for computerization.

There are a number of salient features of STM:

i] *System Definition*

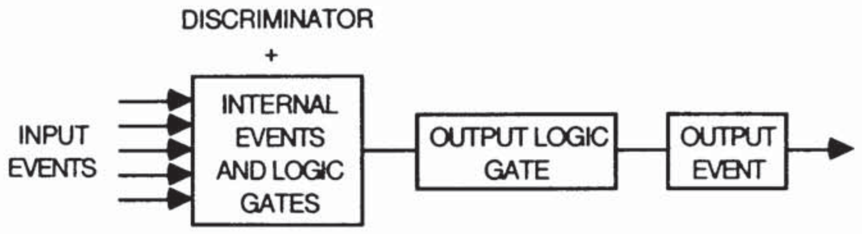
STM was designed to cope with simple linear electrical systems (involving switches, fuses, relays, motors etc). Circuit diagrams showing components and their interconnections are required. Obviously the top event must be defined, but this event may not fall into the class of events defined within STM; here the tree top must also be specified. (For example the event 'vessel overpressure' may have to be refined to the events 'pressure relief switch failed' and 'pump on too long'). To limit the analysis certain system boundary conditions are required. The most significant of these are the top event boundary conditions. These define events that are either allowed or not allowed in the subsequent fault tree (eg for 'pump on too long' a not-allowed event is 'pump stopped', as the two cannot coexist).

ii] *Component Failure Transfer Functions*

These define the causal activity of a component. A set of generic transfer functions is stored in a library, to be drawn on and instantiated for each occurrence of the component in the system under consideration. A component has one failure transfer function for each of its fault and failure modes, Figure 2.5.4a [Fussell 1973a], comprising:

- output event - a particular component failure mode
- output logic gate - determining the logic gate to be used when combining FTFs with the same output event
- internal fault logic in the form of mini-fault trees, with input events in the form of primary failures or command faults
- discriminator - conditions governing the coexistence of minitrees in the final fault tree

FIGURE 2.5.4a <u>Component Failure Transfer Function</u>



iii] *Component Coalitions*

A component coalition is defined as a subsystem containing all those components which lie on a single series circuit path, ie share a single current flow path. This provides for the definition of subsystem failure modes, eg 'no current in coalition X', which may be related directly to corresponding component failure modes, eg 'no current in component x'.

iv] *Fault Events*

Apart from primary failure events, a classification of fault events is suggested:

1st order - used only as top events, these must be refined further to include events specific to components or coalitions in the STM model.

2nd order - fault events occurring at the coalition (subsystem) level, encompassing more than one component.

3rd order - component command faults due to subsystem or coalition failure.

4th order - component command faults due to directly connected component failure.

v] *Event boundary conditions*

These are "allowed" and "not allowed" events associated with a given event when it is added to the fault tree. The domain of an event E is defined to include any event lying in the branch below E in the fault tree. If E forms an input to an AND gate, all inputs to that gate also fall in the domain of E. Two incompatible events for the same component, eg 'fuse blown' and 'fuse overload' cannot coexist in a given domain. Thus when E is added to the fault tree, boundary conditions must be considered when developing the domain of E. "A most important corollary is that a fault event is equivalent to another fault event if, and only if, their event descriptions and their associated event boundary conditions are, in effect, identical" [Fussell 1973a]. This explains the phenomenon where two fault events in a tree may appear to be identical (having the same item and incident definitions), but are not so because their causal structures are different and incompatible. It is clear that 'XXX' caused by 'AAA' results in a different system state than 'XXX' caused by 'BBB'.

Fussell goes into great detail in the definition and classification of boundary conditions for the different orders of events.

vi] *Tree Construction*

An algorithm is used to develop the specified tree top down to primary or undeveloped events. This proceeds according to the order of the event being developed, logic governing inclusion/exclusion of events under AND and OR gates, and existing or generated event boundary conditions. The order of an event determines the use of failure transfer functions or the employment of the direct hierarchical relationship between current flows in components and coalitions.

In attempting to apply component mini-fault tree model based synthesis to chemical processes a major problem is soon encountered - how to link the component models together? Information flow between electrical components is restricted to one parameter - current flow, whilst in chemical processes components may interact through mass, energy or momentum transfer, ie by means of many continuous variables.

Powers [1974,1975,1976]

Powers and Tompkins addressed this problem by devising Information Flow Models for components. Based on rigourous mass, energy and momentum balance equations these models take the form of "gain matrices" defining qualitative relationships between independent and dependent component variables. Different components are linked by those variables which they have in common. In the system as a whole, these component linkages provide "Information Flow Paths" through the system model.

With the use of minitrees to define the 'tree top' for a particular event, subtrees are refined by tracing through the diverging information flow paths. Component failures are described by minitrees for use when a failure mode is required for completion of an information flow path.

This method does not provide a complete fault tree synthesis structure (questions of logic gate selection, control loops and fault tree coherence are not addressed), but rather a means of ensuring that all component interactions are considered. The component models (information flow matrices and failure minitrees) are extremely exhaustive and complex, involving consideration of all the variables associated with the component. Attempts are made to comprehensively classify the information that is required about a system before fault tree synthesis can take place: equipment and chemical property data, along with equipment failure mode data.

In describing component interactions, then, attention is focused on key process variables rather than on the components themselves. In this respect this work can be seen as the precursor to the variable-digraph approach of Lapp and Powers discussed below (2a Variable Digraphs).

Hollo [1977], **Taylor** [1977,1982], **Olsen** [1978,1979]

The work carried out at the RISO Laboratories also uses component models to represent chemical processes in a fault tree and cause-consequence diagram synthesis strategy. Each component has a set of ports; connections between components, via these ports, are defined by the user. For each component there exists a generic functional and failure mode model, stored in a library, consisting of a set of mini-fault trees. These minitrees causally relate component outputs (in terms of variable values and component states) to input conditions and internal failure modes. Fault tree synthesis proceeds from the top event downwards by progressive refinement of the leaves of the tree. Component minitrees are combined by matching output and input events at ports linking components to form the overall tree. In contrast to the Powers and Tompkins method, then, attention is focused on physical rather than parametric interconnections between components, and on events rather than variables as component input/output.

Component models are constructed in a systematic manner to enable them to be combined meaningfully in various system configurations. In a similar manner to that of Powers, qualitative relationships between all component variables are rigorously defined from steady state and dynamic equations. A large number of minitrees (typically 20 to 60) are defined for each component. Feedback loops are catered for by introducing special minitrees for those components which serve to reduce or eliminate process disturbance, in order to introduce AND gates into the final fault tree. These minitrees take the form of "output event occurs if input event occurs AND compensating component fails". Provision is also made for introducing time delays into the component cause-effect models to permit event sequencing, as these models have also been used for Cause Consequence Diagram construction [Hollo 1977] (CCDs are discussed in 2.6).

Martin-Solis [1977,1982]

This work proceeds along similar lines in the synthesis of fault trees. The methodology evolved from the alarm analysis network synthesis [Andow 1975] discussed in 2.7. Component models consist of mini-fault trees derived from functional equations; the mini-trees are combined in the overall fault tree. To ensure coherent trees, use is made of the concepts of domain and event boundary conditions as described by Fussell [1973a] and above (1a).

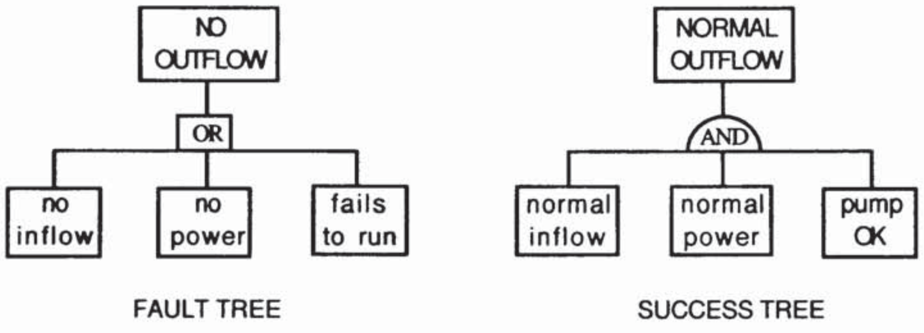
1b) Decision Tables

Wu [1977], **Salem** [1975,1977,1979]

In this representation components are modeled by decision tables [Pollack 1971] which relate each possible output state of a component to a set of combinations of states of inputs and internal operational or failure modes. The decision tables are essentially a compact means of recording the same information that could be conveyed by a set of mini-fault trees (their principal advantage is that of ready storage and manipulation by computer). Consider the simple decision table for a pump, and the corresponding minitrees in Figure 2.5.4b. (Wu et al use a numeric notation for variable values and failure modes which is here translated for clarity). For a given row in the table, the output value is true if all the preceding values in the row are true ('-' for a value indicates "don't care"). This provides for the representation of AND logic between events (required for control loops).

FIGURE 2.5.4b Decision Table and Mini-Fault Trees - a Simple Pump Model

Input 1 - Flow In	Input 2 - Power	Internal Mode	Output - Flow Out
No	-	-	No
-	No	-	No
-	-	Fails to run	No
Normal	Normal	OK	Normal



The coupling of components is achieved by defining a set of 'nodes' throughout the system. The output from each component is connected to a node which is, in turn, connected to the inputs of one or more succeeding components; this provides for divergence in fault propagation. Thus an output event of a component can be said to occur at its output node; this nodal event corresponding with input events of succeeding components.

Tree synthesis proceeds from the initial definition of the tree-top by successive refinement of the tree leaves, tracing from effects to causes through component decision tables and via linking nodal events. The fault tree consists of events occurring at nodes (component input/outputs) along with component fault and failure modes. Logical correctness is addressed by editing during the construction of logic gates using rules defining compatibility of events and event boundary conditions similar to those of Fussell (above).

Kumamoto [1979] present a similar method of using decision tables to model input/output/failure relationships. Synthesis of fault trees is not attempted, however, the goal is the direct production of cut sets for a selected top event. The top event is first defined as a single event decision table. This table is refined by successive replacement of those events in the table which are defined as output events of component decision tables. The end result is a decision table for the entire system (called a critical transition set) with the top event as output, and primary events as inputs. Each row in this final table represents a cut set for the output event. The critical transition table resembles the fault matrices [Lihou 1980e] and system decision tables [Berenblut 1977] which have been used for fault analysis by pattern recognition (section 2.7).

By this side-stepping of the fault tree representation, the aim is to avoid difficulties in logical correctness of synthesised fault trees by not producing any. In practice, however, rules governing transition from one decision table to another, as defined by Wu et al above, must be employed for correct and complete cut sets.

2) GRAPH MODELS

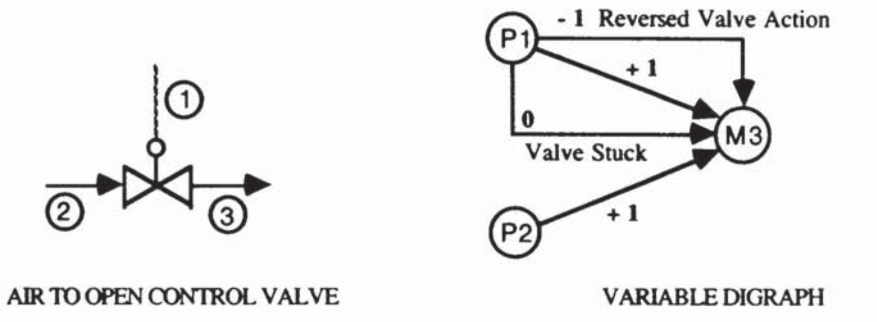
2a) Variable Digraphs

Lapp [1976,77a,77b,77c]

The previous section (1a) discussed how Powers and Tompkins [1974,75,76] devised information flow models for components to describe the directional relationships between component variables. These relationships may be expressed diagrammatically in the form of variable digraphs. Lapp and Powers described the manual synthesis of digraphs for chemical processing systems and presented an algorithm for the synthesis of fault trees from this plant digraph.

Consider the simple variable digraph for the control valve shown in Figure 2.5.4c. The nodes in the digraph represent variables in the streams connected to the devices. The edges show the direction and magnitude of how change in one variable causes change in another, whilst incorporating conditions which are failure modes of the device. For example, an increase in signal pressure (P1) causes an increase in output flowrate (M3). In the case where the valve failure mode is "valve stuck", no change propagates.

FIGURE 2.5.4c Example Component Variable Digraph



P1 = control signal pressure
P2 = pressure - valve input stream
M3 = mass flowrate - valve exit st ream

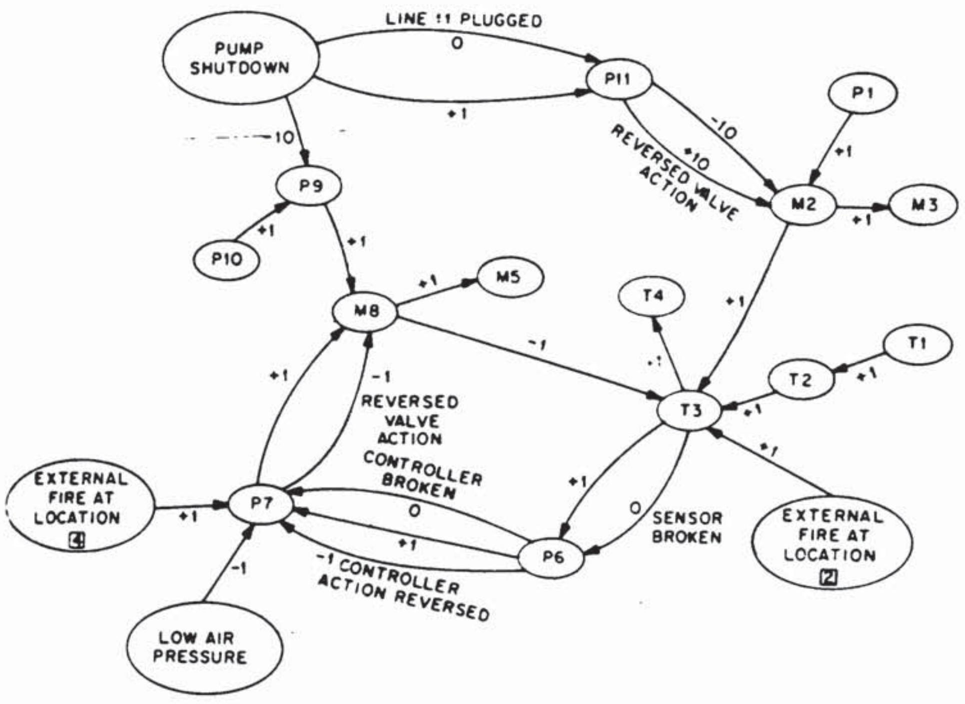
The complexity of the digraphs representing a component depends, naturally, on the number of variables and failure modes considered.

The production of the fault tree for an event involves these stages:

1. Define the system, specifying streams and nodes (P&ID)
2. Define the top event, and the variable of which it is a deviation
3. Construct (manually) a plant digraph for the top event and transfer to computer
4. Automatically synthesise the fault tree from the plant digraph

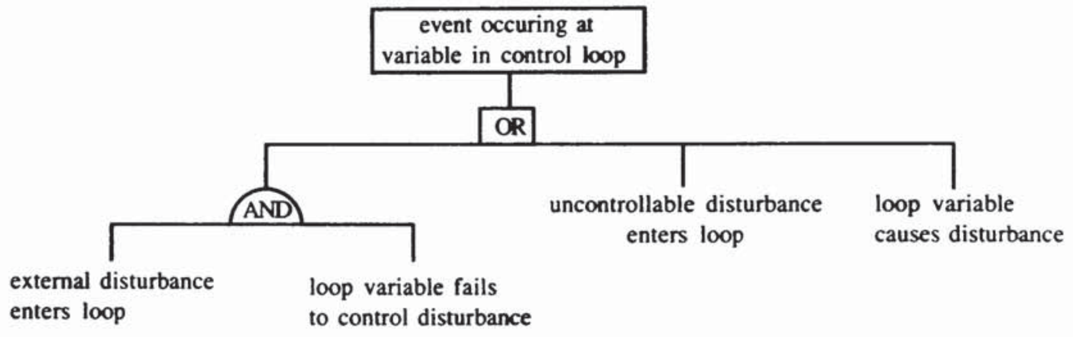
The variables of a given stream in the process may serve as nodes in the digraphs of several components, the component digraphs may then be linked by these nodes to form an overall plant digraph. An example is Figure 2.5.4d showing the digraph for the variable T4 of the Nitric Acid Cooler.

FIGURE 2.5.4d Plant Digraph for Variable T4 of the Nitric Acid Cooler [Lapp 1977]



The plant digraph for a variable may be used to synthesise fault trees for top events which are changes in this variable, proceeding in the standard manner by successive tree leaf refinement. Essentially the method consists of tracing through the digraph along all appropriate paths noting causal changes in variable and failure modes; these appear in the fault tree. Multiple causes for a given variable deviation may be grouped under an OR gate. In some circumstances, however, AND gates must be incorporated; this occurs when encountering control loops in the process. Lapp and Powers used general loop fault trees, programmed into the algorithm, to tackle the problem, Figure 2.5.4e - these trees are used when developing the causes of the deviation of a variable in a control loop. Process loops are readily detected from the structure of the plant digraph.

FIGURE 2.5.4e Generalised Fault Tree for a Feedback Control Loop


event occuring at variable in control loop
OR
AND
uncontrollable disturbance enters loop
loop variable causes disturbance
external disturbance enters loop
loop variable fails to control disturbance

The problem of consistency checking illustrates the general principle assumed when defining an event in a fault tree, namely that no other events or failures occur in the domain of the event that are contrary to its causes and effects. L+P proposed the use of exclusive-OR (XOR) gates to be included in the tree when one and only one of the input events being true causes the output event to be true. By the introduction of NOT logic this violates the independence of primary events leading to incoherent cut-sets. The use of NOT logic (XOR is equivalent to a combination of NOT, OR and AND) gave rise to considerable debate [Locks 1979, Yellman 1979, Lambert 1979, Lapp 1979]. The general consensus seems to be that XOR gates are not necessary as the principle "all other things being equal" is generally true; it is usually safe to assume, when considering a given event sequence, that the rest of the plant operates normally. In other words, OR gates are sufficient.

The Lapp-Powers algorithm has been applied to electrical systems [Cummings 1983]. These systems are considerably easier to analyse than chemical processes for a number of reasons:

- Many devices are 2-state: either working or failed. CP devices and equipments have multiple failure modes.
- There are only two variables - voltage and current, which may be treated digitally, ie either present or not. Process systems have many interlinked continuous variables.
- Synthesis of the variable digraphs is easier because of the above and because component digraph models are simpler and involve little complex variable interaction.

The successful application of the technique to electrical systems (plus more recent refinements) does support the thesis of Lapp and Powers, i.e.: providing that the digraph is a complete and correct representation of system causality then the fault tree can be synthesised from it, using appropriate prior generalised fault trees for control loops. Attention is then focused on the production of plant digraphs from component digraphs and this is where the main problems of the method lie. More recent efforts have refined the algorithm for synthesizing fault trees from plant digraphs [Allen 1980, Andow 1980a, Ulerich 1988] but none of these address this central problem; automated construction of plant digraphs from component digraphs and a system description has not been demonstrated. Synthesis of variable digraph models for complex equipment is very difficult. It appears that system digraphs were constructed by hand for each top event as the structure of the required digraph is dependent on the domain of the specific top event selected. Automated synthesis of digraphs has yet to be demonstrated. Construction by direct combination of component digraphs cannot be relied upon to produce complete and correct plant digraphs.

2b) Signal Flow Graphs [**Aggarwal** 1978, **Kumamoto** 1981, **Misra** 1970]

Signal flow graphs are similar to variable digraphs in that they consist of nodes representing process variables, and directed edges representing variable interactions. Each edge, however, has only a transmittance (or gain) associated with it; primary failures are modeled by their effect on source variables added to the signal flow graph. Edge gains have continuous values derived from mathematical approximation of plant operating parameters and gains may be summed over a given node. A top event is defined in terms of an inequality in a variable value and graph theory is used to derive causal source variable inequalities and thus causal primary failures.

This method is unsuitable for automatic fault tree synthesis as it has all the disadvantages of the variable digraph method along with those of its own. It is not feasible to effect the automated synthesis of the signal flow graph from the process description and individual component signal flow models. Whilst it is possible to define control systems rigourously in terms of information flow this is extremely difficult, if not impossible, for complex equipment.

2c) Reliability Graphs

Camarda [1978] has described an algorithm for the automatic synthesis of fault trees from a reliability graph. This method is not suitable for chemical processing systems as the reliability graph is a 2-state representation - components are considered as either working or failed, ie multiple failure or fault modes are excluded.

Reliability graphs are, however, finding increasing use in the evaluation of network reliability for communication systems, eg telephone or computer networks, as these systems are readily defined in terms of information flow. It is interesting to note the general idea of synthesizing system failure modes from a graphical description of all the ways in which the system can operate successfully. This is advantageous because the ways in which a physical system can operate are much fewer than those in which it can fail. Indeed, for 2-state systems, it is possible to directly transform the success tree (a fault tree type structure whose events are successes) into the fault tree by event inversion and the simple exchange of AND for OR and vice-versa [Joller 1982b]. The fault tree and the success tree in Figure 2.5.4b illustrate this.

The overall problem characteristic of graph based methods seems to be that it is at least as difficult to synthesise the graphs as the fault tree itself, so if the graph has to be produced manually, why bother?

3) STRUCTURAL MODELS

In all the methods of fault tree synthesis for chemical processes described so far, a central problem that has to be addressed is that of successfully modelling the action of control loops under fault conditions. It is not possible to do this by qualitative component modelling only, and all the methods rely on previously defined generalised fault tree templates to be applied in each case, for example the generalised tree of Figure 2.5.4f. Automatically synthesised fault trees also tend to be ill-structured and rather opaque to the user. In order to address these problems directly, Shafaghi et al [Shafaghi 1984] devised an approach where the primary decomposition of the plant is not into components, but into control loop structures. The plant is considered as a series of control loops incorporating process equipment. Connections between control loops are defined by considering the outputs of one loop to be the inputs to others - this forms a control loop digraph, whose nodes are loops and edges are loop interconnections. The method consists of the following steps:

1) Definition of the plant control loop digraph.
2) For a given event, generation of a generalised fault tree from the loop digraph.
3) Substitution of the elements of the generalised fault tree by a specific failure structure for each loop. In essence these structures consist of sets of fault trees representing all fault propagation within the boundaries of the loop.

The method as described has a number of shortcomings:

a) The plant description is a control loop decomposition which must be manually defined from the process flowsheet (eg P&ID).
b) Control loop failure structure models, whilst having generalised properties (all negative feedback loops have the same basic structure), must be carefully constructed for each loop in the plant. This is necessary to facilitate the inclusion of the failures of the differing equipment which may be encountered in each loop.
c) Only process deviations which are directly related to control loop action may be included in the fault tree.

This method, on its own, is not a complete method of producing fault trees for chemical processes by computer. External fault propagation information is required to complete the loop failure structure models, and to "fill in the gaps" where faults are not directly concerned with control loops. By constructing fault trees whose skeleton is first defined by the plant control loops structure, however, the trees have a consistent structure and are readily understood. The method emphasises the fact that control loops can only be handled adequately by considering them in their entirety, rather than as a collection of individual elements. With the inclusion of component models to "fill in the gaps" of causality and to assist in the definition of loop failure models, an overall strategy for fault tree synthesis emerges: it is necessary to combine both component level and structural level information in order to produce complete, correct and well-structured fault trees.

2.5.4.4 <u>Discussion of Synthesis Methods</u>

In spite of the efforts detailed above, automated FTS has met with limited success and has not gained widespread acceptance. (At least, outside the electronics industry, in which the systems are much easier to analyse and the requirement is often for a means of testing or a strategy for fault diagnosis). A general reason for this lack of success is that chemical process systems are simply too complex for a rigourous algorithmic analysis: one might as well attempt to construct numerical simulations. In attempting to tackle the problems of complexity one is forced into investigation in more and more detail, throwing up new and even more intractable problems. There is a subtle paradox at work here:

1) It is not possible to analyse all the complexities of a system without considering some of the detailed interactions of its components

2) It is not possible to evaluate component interactions reliably without considering the system as a whole (cf the dynamic behaviour of control systems)

The inescapable conclusion is that both levels of analysis are necessary.

Problems with digraph models include:

1) The task of preparation of the digraphs often equals that of FTS by hand

2) The digraph essentially represents normal information flow paths, where failures propagate from variable to variable along these previously defined paths. It is not possible to synthesise, from the plant digraph, those system failures which involve changes in the <u>structure</u> of the digraph, eg where there is contamination of one stream by another, or an abnormal flow situation occurs.

3) The digraph required depends on the system fault of interest. Unless all plant variables are included in digraphs, this limits the analysis to a single, selected top event.

4) Difficulties in evaluating control system responses are overcome only by embedding structural rules in the synthesis algorithm.

Andow [1980a] has discussed the problems associated with component models based on pressure/flow relationships which are used in the literature. In order to be complete, these models approach the complexity of mathematical models, indeed some of the methods involve compilation of component models from the sets of steady-state and differential equations describing their behaviour. This is clearly not practical for equipment of any great complexity if the general applicability of the causal models is to be maintained. Moreover, fault trees, or other causal structures, do not require the elucidation of a multitude of detailed events; only the set of credible and significant events is needed, and, in principle, one is only interested in the direct causal relationships between these events.

Problems with all automated FTS methods include:

1) The synthesised fault trees are unlike any that would be constructed by hand.

2) No always-satisfactory algorithm has been produced to deal with control loops. It is always necessary to consider the loop as a whole rather than as a collection of components.

3) Two-way flow of information is difficult to model, eg flow disturbances propagate both upstream and downstream from a line blockage or valve closure, and in some cases, may cause looping.

4) The time dimension is often ignored - except in the work of Taylor and Hollo [Taylor 1977, 81b] in producing Cause-consequence Diagrams and Event Trees. Time obviously plays a part in considering batch or sequential processes, which none of the above methods do.

5) None of the published methods comes anywhere near representing the behaviour of complex equipment, eg distillation columns or reactors.

6) The capability to deal with a large number of different systems within a given domain (especially process engineering) has yet to be demonstrated

7) There is a confidence problem - a fault tree from a "black box" is an unknown quantity. Although computerization permits a more exhaustive study, an important feature of manual fault tree synthesis is that it provides the analyst with an intimate knowledge of the process. The analyst's expertise determines the completeness of the resulting study. For this reason any synthesis methods are likely to remain as aids only.

8) The application of automated FTS to chemical process systems has additional problems compared to the application to electronic systems:

   (i) Process variables have a continuous range of possible values, in contrast to the binary nature of many electronic systems.

(ii) Equipment may have a continuous range of degradation and multiple failure modes, rather than two states of operation and failure.

(iii) The actions of process operators may need to be included.

Clearly, fault propagation analysts do not operate by exclusively considering either digraphs, component models or structures. Considering a system at the minimum appropriate level of detail is all that is necessary; the use of structured models incorporating various levels of abstraction seems to be the way forward.

Due perhaps to the failure of automated synthesis to gain widespread acceptance, in recent years efforts have been directed more towards computer based decision support systems - Fault Diagnosis (FD) and alarm analysis systems - for use in high-risk enterprises. As these diagnosis systems employ causal logic models - often directly using fault trees or alarm networks - there is some common ground between FTS and FD. This is particularly the case where the diagnosis system uses structural models to generate and employ causal models. This shift of emphasis is illustrated in the employment of variable digraphs for FTS [Lapp 1976, 1977b] and for FD [Ulerich 1988]. These fault diagnosis / alarm analysis systems are discussed in section 2.7.

2.5.5 Fault Tree Evaluation

Lee [1985] presents a useful review of fault tree evaluation methods which may be divided into the qualitative and the quantitative.

1) Qualitative Evaluation [Allan 1981, Bengiamin 1976, Garriba 1977, Nakashima 1979, Rasmuson 1978, Wagner 1977, Wheeler 1977]

Visual inspection of a fault tree can give valuable insights into the fault propagation characteristics of a system, but the most common qualitative evaluation involves the production of the minimal cut sets (MCS) of the tree. A minimal cut set is a set of primary failures which are both necessary and sufficient to cause the top event in the tree. The enumeration of all the MCS gives all possible combinations of primary events which can cause the system to fail. This enumeration is achieved by logical expansion of the fault tree to give cut sets, followed by elimination of redundancy to give the minimal cut sets. For complex trees there may be a large number of MCS; many algorithms for efficient enumeration were developed in the 1970s [Garriba 1977, Nakashima 1979, Rasmuson 1978, Wheeler 1977], but the problem has been somewhat alleviated by the increased computer power available in more recent years.

A minimal path set or tie set is a set of primary events whose non-occurrence ensures the non-occurrence of the top event. The MPS are found by constructing the dual fault tree (sometimes called the success tree), by substituting AND gates for OR gates and vice versa, and replacing each event by its complement. (The complementary event to a component failure is component success; strictly speaking this is applicable only to 2-state components, whereas chemical process components have multiple failure modes. The approximation is justified where the probability of each failure mode is small.) The minimal path sets of the fault tree are given by the minimal cut sets of the success tree.

2) Quantitative Evaluation [Fussell 1976, Laviron 1982, Rosenthal 1980, Wheeler 1977]

Quantitative evaluation involves calculation of the top event probability based on primary event failure data. The assembly of reliable failure data often presents major problems, particularly with regard to human error, but these problems are outside the scope of this work.

The first step in probability calculations is often the production of the minimal cut sets of the tree. The probability of the top event is then the disjoint sum (given by the OR relationship) of the probabilities of the MCS. Alternatively, probabilities may be calculated recursively in the tree, working from primary events upwards.

A major problem occurs when, as is most often the case with systems of any complexity, there are repeated events in the fault tree [Bengiamin 1976, Allan 1981]. Here a straightforward calculation is invalid, as it relies on the assumption that all events are independent. An event may be repeated in a tree where it may combine with other events to lead to the top event via a number of different fault propagation routes. This, in fact, is common cause of intermediate events; straightforward calculation of the top event probability is then incorrect because the repeated event makes a multiple contribution. Mutually exclusive events (occasionally associated with NOT or XOR gates) or any event dependency leads to the same problem There are various approaches to the problem of repeated events including Monte Carlo Simulation, Bayesian approximation [Jones 1987] and Disjoint methods. The scale of the problem is, however, an exponential function of the number of repeated events.

Whilst the current work is not directly concerned with fault tree evaluation, if fault trees are to be produced via HAZOP equations then qualitative analysis is useful as an aid to evaluating the correctness of these trees. This correctness may be judged by the correctness of the cut sets derived from the synthesised trees.

2.6 OTHER CAUSAL REPRESENTATIONS [Andow 1980b, ISGRA 1985]

In order to identify, describe and evaluate system failures, a number of different techniques have evolved. These techniques are all concerned with the analysis of fault propagation and as a result they are similar in their nature and in their problems. They may be considered to operate either deductively, inductively or as a combination of the two:

Inductive - proceeding from cause to effect - identifying the effects of a particular event

Deductive - identifying the causes of a given event

The techniques of HAZOP and Fault Tree Analysis are discussed elsewhere (2.4 & 2.5)

Event Trees [Henley 1981, Heslinga 1983, Taylor 1981b]

The specific mode of failure of an item of equipment is selected. The potential courses of events that might follow, as the fault propagates through the plant, are developed. The result is similar in appearance to a fault tree, but proceeds from a single primary event through to several possible consequences, via success or failure of intermediate events.

Failure Modes and Effects Analysis [Browning 1970, ISGRA 1985, Recht 1966a]

FMEA involves consideration of the effects of all critical component failure modes:

1) Identify and specify all component failure modes of interest
2) For each failure mode, identify effects on other components and on the performance of the system as a whole

3) Estimate the seriousness of the consequences of each failure sequence

In order to identify every system failure which is caused by a combination of events it would be necessary to consider all (non-conflicting) combinations of all the identified failure modes. This is clearly not feasible, so in practice reliance is placed on the expertise of the analyst in discovering all the significant failure combinations. FMEA does not permit reliable probability calculations as there is no guarantee that all failure combinations have been considered. It is possible, however, to combine the probability of occurrence of each failure mode with a measure of its severity to provide a ranking order. This is known as Failure Modes Effects and Criticality Analysis (FMECA).

Cause-Consequence Analysis [Nielsen 1971,75, Taylor 1981b]

A critical or potentially hazardous event is first selected. The consequences of this event are considered, along with subsequent equipment and personnel action or failure. The causes of the critical event and other equipment failures are included as fault trees. This represents, in effect, a combination of fault trees and event trees to give an overall picture of the critical event: its direct and contributory causes and its possible consequences. The method also has the advantage in that it takes into account, where appropriate, the time ordering of events.

Alarm Analysis

Alarm trees are computer data structures which are used to represent the different sequences of alarms which may occur in a process. When an alarm sequence does occur, the representation may be of assistance in diagnosing the problem and identifying the failure that initiated the sequence. The use of alarm trees and networks for the real time analysis of fault propagation in process plant is discussed in 2.7.

As may be seen from the above descriptions:

- FTA is a deductive technique, for identifying event sequences which can lead to a particular outcome, particularly useful in assessing the likelihood of that particular outcome.
- FMEA and Event Trees are inductive methods, used primarily as a means of evaluating the system reliability implications of critical components.

- HAZOP and Cause Consequence Diagrams express both the causes and the consequences of the selected events. The former assists in the identification of undesirable outcomes whilst the latter provides a comprehensive picture of the circumstances surrounding a particular event.

Each of these techniques is of value in the appropriate circumstances. To illustrate:

- A HAZOP identifies a serious hazard
- Fault Tree Analysis evaluates the likelihood of the hazard and reveals that a number of components are critical to successful operation
- FMEA and Event Trees develop fully all the consequences of failure of the critical components
- Cause-Consequence Diagrams are used to present the overall picture of the hazard

It is evident that there are strong similarities between the different techniques and their representations; the analysis of fault propagation in process plant is a generic problem [Andow 1980b]. A full discussion of the knowledge used and the reasoning employed may be found in Chapter 5.

2.7 FAULT DIAGNOSIS

2.7.1 **Introduction**

An essential requirement for the successful operation of a chemical plant is the ability to detect process deviations and fault conditions, to determine their causes and possible consequences, and to take remedial action soon enough to prevent loss. To this end, one may rely on:

- Human ingenuity, intelligence, training and experience. [Pau 1981, Rasmussen 1978,81, Su 1986]
- Computer Aids for assistance in fault diagnosis, consequence prediction, remedial action and consequence mitigation (discussed below).
- Diagnosis according to a sequence of actions worked out in advance and intended to minimise the time taken to locate and correct the fault [Lihou 1980a,e].
- Go to bed and hope everything turns out all right in the morning.

Of relevance here are computer aids for fault <u>diagnosis</u>. The term diagnosis here refers to the process of deducing, from a set of observed plant states, the primary equipment or environmental (in a direct sense) causes of the observed states. The requirement is naturally different from fault <u>analysis</u> in that one is attempting to discover a particular fault condition according to a set of observations in real time, rather than to describe fully, at relative leisure, the different fault sequences that may occur in a particular design. For example, in diagnosis it is common to make the assumption that plant deviations are due to a single initiating event, rather than a (generally less likely) combination of causes.

This discussion includes the subject of alarm analysis, a technology developed for reducing the apparent complexity of multiple alarm displays, particularly with regard to modern computer controlled plant. This involves determination of groupings and relationships between alarms and the presentation of this information to the operator. It has found widespread application in the nuclear industry, where the number of potential alarms may run into the thousands [Andow 1978].

These systems are often based on a previous fault analysis such as a HAZOP or a Fault Tree Analysis, but some of them attempt to derive causal behaviour from a structural representation of the plant. The differing approaches to the problem may be seen from the literature overview of Fault Diagnosis and Alarm Analysis given in Figure 2.3.1f.

2.7.2 **Outline of Computer Models**

In considering the different generic aspects of computer aids it is possible to divide them into two loose categories, according to the basic model of the process used by the computer. This is generally either a causal model or a structural/functional model (Rasmussen makes a distinction between structure and function [Rasmussen 1979a], but the assumption in most of the models is that function is implied by structure and therefore implicit in the representation). A distinction is also made between algorithmic and knowledge based systems. The term algorithmic refers to "conventional" software, data structures and programming methodology, distinct from Knowledge Based, "Expert", or AI systems as discussed in Chapter 3. This distinction is most apparent in the representation of the system under analysis, contrasting "conventional" data structures with networks, rules, frames, constraints etc. Pau [1986] presents a review of IKBS approaches to fault diagnosis, test generation and maintenance problems.

Pattern Recognition

Early attempts at computer aided fault diagnosis made use of recorded characteristic effects of faults in a particular plant. For a given fault, the symptoms of the fault will become apparent to the operators as changes in certain observable variables, ie these variables deviate to high, low etc. These relationships between a fault and its symptoms may be stored in a computer in the form of matrices [Lihou 1980e] or decision tables [Berenblut 1977, Munday 1977]. When deviations in process variables are observed, the set of deviations may be compared against those stored in the tables or matrices by a process of pattern matching. When a match is found the existence of the corresponding initiating event is implied.

These systems work by a process of simple pattern recognition. The plant itself is considered as a "black box" with inputs (initiating faults) mapped to outputs (Process variable deviations). In order to construct this mapping a previous fault analysis, usually HAZOP is required. In the case where the HAZOP has been recorded as Cause and

Symptom Equations [Lihou 1980e] which are described in Section 2.4.5 the equations may be directly translated into plant state matrices.

Causal Models

These models involve interpretation of observed symptoms according to a pre-defined model of fault propagation developed for the particular process. In other words, the computer stores the complete results of a previously conducted fault analysis, eg HAZOP or FTA. This information may either

1) Directly relate observations to primary causes or final consequences.
2) Represent detailed cause-effect relationships relating primary causes to ultimate consequences via fault propagation pathway, (either conventionally, or as part of a KBS)

Some of these methods, which use an exclusively causal model, attempt the automated synthesis of the causal model using methods developed for Fault Tree Synthesis, for example from variable digraphs [Ulerich 1988] or component models [Andow 1975].

Structural Models

These methods involve the interpretation of symptoms in the context of a structural or functional model of the plant which incorporates the causal behaviour of the components, from which causal hypotheses may be constructed. The model is initially defined by a structural description of the system. This is combined with a behavioural description of the components or substructures contained by the system. The behavioural descriptions may be generic in nature being instantiated according to the particular components in the system of interest. In this way the method resembles those used for the synthesis of fault analyses, in that a structural description of a system is combined with causal descriptions of its components in order to describe the causal behaviour of the system as a whole. The advantage of this approach is that one need only describe the structure of the system to the computer for each case, rather than having to conduct and describe an *a priori* fault analysis. Again, conventional approaches have more recently given way to IKBS methods, the synthesis of fault analysis by IKBS being of direct concern in the present research.

2.7.3 **Causal Models**

We can distinguish three types of causal model:

1) Direct relation of observations to primary causes or ultimate consequences. This operates via straightforward pattern matching of observed symptoms or event sequences to primary causes or possible consequences, the symptom to cause or consequence mapping is drawn from a previously conducted analysis (eg HAZOP or FTA). The patterns may be stored as:

   Event Sequence Records - derived from previous incident records or *a priori* analysis.[Baybutt 1986].

   Decision Tables - each decision table rule consists of a set of possible observable process variable values which are indexed to a possible cause for diagnosis [Berenblut 1977, Himmelblau 1978], or consequence for deciding if the observed conditions will lead to an undesirable outcome [Munday 1977].

   Plant State Matrices - A state matrix of process variables is formed from on-line observations and compared with previously defined fault symptom matrices. A positive match implies that the fault is the one defined by the matched fault symptom matrix. This technique is useful for diagnosing the causes of disturbances in complex equipment [Lihou 1980e].

2) Conventional computer representation of fault propagation:

   Fault Trees - in this approach a predefined fault tree of a particular event is used to determine its possible causes, usually by enumeration of candidate cut sets. The fault trees may be generated off-line by computer, Ulerich [1988] uses an updated version of the Lapp/Powers variable digraph method, described in 2.5.4.

   Decision Tables - The information conveyed by a fault tree may be stored as decision tables and used in a similar manner, ie cut set analysis [Berenblut 1977, Himmelblau 1978].

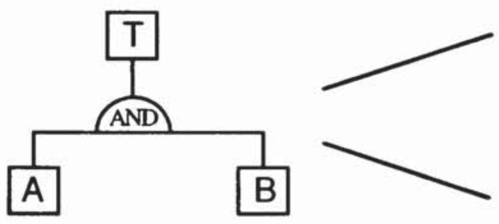
   Alarm Trees - networks representing groupings between alarms and the cause-effect links between them [Andow 1974, 75, 78].

   Variable Digraphs - using a variable digraph of normal links between process variables to determine causes of variable deviations [Tsuge 1985b].

3) Representation of fault propagation using IKBS.

These systems represent causal information in a data structure, and using a programming language that allows for IKBS methods. In practice most of them essentially represent similar information to that contained in a fault tree, but encoded in the form of rules as shown in Figure 2.7.3a [Andow 1985a, Chester 1984, Kumamoto 1984].

FIGURE 2.7.3a Encoding a Fault Tree as a Rule



IF	A and B	(exist as events)
THEN	T	(will occur)
IF	T	(occurs)
THEN	A and B	(is the cause)

Variations:

- IF <system in state1> AND <observa ble fact> THEN <system was in stat e2>
- IF <component x OK> OR <component y OK> then <component z OK>
- category <event> has subcategory < subevent> provided <fact>

Rules may also be used to represent more subtle interdependencies between states, eg consistency between measured variable, indicator status and indicated value.

Storing the information as rules allows for the use of knowledge engineering techniques such as search space strategy, application of heuristics, information request selection, weighted inference, explanation facilities etc. The starting point is a set of observed conditions. The inference structure formed by the rules is then searched to establish causes and consequences using forward chaining, backward chaining or a combination of the two. This method of finding candidate sets of initiating failures is very similar to determining the cut sets of the corresponding fault tree. However, if one were to set out to determine exhaustively all the candidate sets for a system of any complexity, the search space: the number of candidate sets to be checked out during diagnosis, would be too large. For this reason a pseudo-probabilistic method (often based on Bayesian inference) is used, for example, breadth first search with 'best option' likelihood filtering based on previously defined penalty functions for each causal path [Andow 1985a]. The 'single fault' assumption is usually made in order to eliminate candidates as the search progresses.

An alternative approach is that of semantic network based representation. Underwood [1982] uses the Common Sense Algorithm (CSA) which was developed in the AI community for reasoning about physical systems, as a basis for problem solving and language comprehension [Rieger 1977]. A causal semantic net is constructed linking CSA-events, describing actions, states and statechanges, with CSA-links, representing causation and interdependence. Diagnosis proceeds from the identification of process disturbances in the network via search strategies to determine candidate causes.

2.7.4 **Structural Models**

The algorithmic systems which rely on an initial structural description and combine this with predefined generic causal models use techniques similar to those for automated fault analysis. For example, Andow and Lees [1975, 78] have used plant unit models to construct fault trees and alarm networks for fault diagnosis and fault analysis.

Particularly in the field of digital electronics it has proven readily feasible to construct generic models of system components. This is due to the deterministic and binary nature of these systems, which are considerably simpler in their behaviour than chemical process systems.

The DART project - Diagnostic Assistance Reference Tool [Genesereth 1982] is designed to pinpoint failed components in computer hardware, on a Least Repairable Unit (LRU) basis. The representation of the equipment to be diagnosed comprises:

- a hierarchical design description at multiple levels of abstraction
- functional models of primary components (eg logic gates) which are designated as LRUs

- behavioural (i/o) models of subsystems (eg a multiplier or an adder).

Diagnosis proceeds from an initial unwanted symptom of the system as a whole (ie the faulty computer board) by generating input/output tests to discriminate between subsystems. This identifies the failed subsystem which itself is then subjected to the same test/discriminate process.. Discrimination of subsystems proceeds in this fashion "down" the hierarchy to narrow down the options to a single LRU (the single fault assumption is made).

DART and other similar systems such as IDT [Shubin 1982] border on the conventional/IKBS boundary in that they use symbolic representation and inference. The CRITTER system [Steinberg 1982] reasons about digital hardware in a similar manner, at multiple levels of abstraction, using constraint propagation [Sussman 1980] to relate behaviour to specifications, although this system is essentially a simulator, useful in both design and the development of testing strategies.

None of the knowledge based systems encountered by the author which analyse fault propagation in physical processes rely on the actual synthesis of causal behaviour from a description of the structure of the physical system [Fink 1985, Maclean 1983, Matsumoto 1985]. A description of the physical system is used rather as an adjunct to a more general experiential knowledge base consisting of problem specific cause effect descriptions; these may take the form of rules as described above.

The reason why a structural or functional description of the physical system is necessary is described by Fink et al [Fink 1985] in their design for an Integrated Diagnostic Model. We may distinguish between two types of problem-solving knowledge: Shallow or experiential knowledge, rules of thumb or in IKBS parlance Heuristics; and Deep or compiled knowledge, that relating to the underlying theory of the domain which enables an expert to reason and explain, where necessary, from first principles. As may be seen in Chapter 5, for an expert system tackling the complexities of chemical processes to be at all robust or generally applicable it is absolutely necessary for both types of knowledge to be represented and used.

2.7.5 **Discussion**

The problems of fault diagnosis and of fault synthesis are similar as in both cases the analyst is attempting to determine the causes and consequences of process deviations, either in real time as an operator, or off-line as a HAZOP or Fault Tree analyst. Therefore one would expect the methodologies, their problems and solutions, to be similar in nature.

As has been stated, the methods of particular interest here are those which involve IKBS techniques for manipulating both causal and structural knowledge about a system. These are readily illustrated by the representation systems used, these are discussed in detail in the IKBS Literature Review in Chapter 3:

Rule Structures - able to represent naturally cause-effect relationships between events and variables. Of importance here is the fact that the rules each represent a chunk of knowledge, which is complete and consistent in itself but which does not necessarily have to be consistent with the rest of the knowledge base to be of value. An analogy would be the use of a set of mini-fault trees for fault diagnosis. If one were to combine all the minitrees one would not necessarily have a complete and correct fault analysis, yet, in diagnosing a particular fault, they provide valid and useful causal paths for investigation. Rule based systems work in an inferential rather than a deterministic fashion.

Networks - Representing physical, causal and interdependency links between components, variables, states, actions, statechanges etc

Frames - Frame based representations of physical objects, their attributes and relations have been widely used. The collection of all the information about a given entity "in one place" is their principal advantage.

Decision Support Systems

One or more of these types of representation may form part of a Decision Support System, of which a number have been developed, particularly in the nuclear industry. These systems generally consist of structured databases containing information regarding accident sequences, causes, consequences and remedial actions [Baybutt 1986b]. More recently there have been attempts to include this knowledge as part of an expert system environment in order to assist in the selection of the particular information required by the decision maker [Embrey 1986b, Williams 1983]. The motivation behind DSSs is the facility to have all relevant information at hand in a readily accessible form, with assistance from the DSS in finding the information required at any particular time.

The FALCON system for rule based alarm analysis in process plant combines the use of rule-based causal representation with a model of the interactions between process variables (stored as transition tables) to deduce causal hypotheses [Chester 1984]. Real time monitoring of a process is used to give an indication of system faults. Whilst systems like this use both shallow and deep knowledge, both types of knowledge bases are developed for each particular physical system. What is required of a truly general Knowledge Based System for Fault Analysis is the ability to produce causal knowledge for a variety of different process plant configurations.

CHAPTER THREE

LITERATURE REVIEW - Knowledge Based Systems

3.1 INTRODUCTION

The technology of Knowledge Based Systems (KBS), sometimes referred to as Expert Systems, is a very young one. It has been made possible primarily by the availability of cheaper computing power in the 1980s. These systems rely on the manipulation of symbols and concepts rather than the numerical manipulation of conventional data processing applications.

This chapter presents a Literature Review of Knowledge Based Systems. An historical perspective is presented which places the technology of KBS in the context of Artificial Intelligence (AI) research from which it emerged. This is followed by a discussion of the tools - computers and software - used for Knowledge Based System building. The bulk of the Chapter is concerned with a detailed description of KBS technology and AI research as they apply to reasoning about Engineering Systems.

3.2 HISTORICAL PERSPECTIVE

This historical review is a brief one and is intended only to place modern work on Intelligent Knowledge Based Systems (IKBS) and Expert Systems (ES) in the historical context of the Artificial Intelligence (AI) techniques which made it possible. Some of the more important historical landmarks help to establish this perspective.

The field of AI and its derivatives is a relatively new one; this means that it is difficult to judge the current relevance of published work by its age alone; relatively old works still have unexplored ideas. It is only in the last decade that economic advances in digital technology have made possible the widespread development of IKBS, so many of the quoted publications date from the 1980s.

Although the idea of a computing machine had been around for a long time, it was in World War II that the first programmable computers were used. These electromechanical devices made a great contribution to the Allied war effort; their contribution to British codebreaking activities significantly altered the course of the war.

A major breakthrough came about when the thermionic valve was superseded by the transistor which was invented in 1947. This made possible, in the mid 1950s, the storing of a program rather than just its data in dynamic memory, giving fast access and facilitating modification of the programmer's instructions to the computer. This led to the first use of high level languages; interactive rather than batch development of large programs became possible.

The first "intelligence" work performed on computers was in game-playing, eg chess and tic-tac-toe (Noughts and Crosses). The term AI was first used in 1958 by John McCarthy at a conference discussing Machine Intelligence. In 1958, McCarthy developed LISP from previous list structure based research, basing his work on the Lambda Calculus for symbol manipulation, which defines the pure language of LISP [McCarthy 1978]. The language rapidly became the most important in AI, particularly in the USA, which, almost alone in the world, devoted extensive resources to the development of AI applications, particularly in the military sphere.

In 1963, the first published volume of AI papers appeared [Feigenbaum 1963]. Many important developments were made at MIT in the Semantic Information Processing projects, illustrated by a published volume of the same name [Minsky 1968]. The use of the word semantic emphasises the focus on meaning in the several models of intelligent behaviour that were being investigated, for example, a formal logic theorem process. Later in the 1960s it was realised that large amounts of background knowledge were needed in any model which competently tackled a particular task. Rather than using formal logic, people reason at a higher level of abstraction, using heuristics or rules of thumb. This emphasis on task competence and background knowledge, defined as heuristics in narrow domains (fields of expertise), gave rise to the first knowledge-based systems. Rather than attempting to simulate intelligence through general problem solving, theorem proving and game playing, more emphasis came to be placed on tackling problems such as image processing and natural language understanding, which, to a human, may not seem to require a high degree of intelligence. (Interestingly, Chomsky has proposed that humans may possess in-built or "programmed" mental faculties for these tasks.)

Progress in the 1970s was somewhat slow until later in the decade when Very Large Scal Integration (VLSI) chips became cheaper and powerful computers became more widely available. At this time the first major successful expert systems projects came to fruition, in the analysis of the structure of organic molecules (DENDRAL [Lindsay 1980]) configuring computer systems (R1 [Rosenbloom 1980]) and in medical diagnosis (MYCIN [Clancey 1981]).

All these initial successes occurred in the United States due to the considerable investment in large-scale AI during the 1970s. In the UK, however, AI research efforts were almost wiped out following the publication, in 1973, of the highly unfavourable Lighthill report to government [SRC 1983]. With hindsight this "influential and highly damaging, but also naive and misguided" report can be seen to have considerably underestimated the potential of the then largely theoretical AI research effort "on the basis of what appeared to be partially understood technical arguments" [Bremer 1988]. There was also an apparent attitude problem: "part of the stimulus to laborious male activity in creative fields of work, including pure science, is the urge to compensate for lack of the female capability of giving birth to children.... a relationship which may be called pseudomaternal rather than Pygmalion-like comes into play between a Robot and its Builder" [SRC 1973]

Whilst some research continued at Edinburgh and Sussex Universities, the influence of this report on the levels of investment that industry and government were prepared to make are still in evidence today; small, PC-based systems performing largely trivial tasks are common in the UK whereas in the US larger, workstation based systems of greater individual significance tend to be the norm. The Alvey programme of the 1980s has been seen by many to be too thinly resourced to produce the hoped-for boost to UK expert systems technology.

The continuing economic development in digital hardware has produced an astonishing fall in the price of computers in the 1980s. This has facilitated the rapid upsurge in IKBS development seen in recent years. There is now a plethora of highly successful, but small, "Expert" Systems working in restricted domains. These are mainly in-house systems designed to tackle specific problems, and are usually implemented using PC-based expert system shells such as Xi-Plus. In developing systems of any great ability, however, large amounts of computing power are still required and for this reason personal computers are not considered adequate. Purpose built AI workstations are proving invaluable in dramatically reducing development time by providing a complete development environment.

Although LISP is still the principal language for AI work in the USA, the announcement by the Japanese that they were to use PROLOG in their ambitious 5th generation project led to a surge in the use of this logic programming language, which has always been popular in the UK. There are now a large number of alternative, but largely derivative, symbolic programming languages available (eg LOGO, POPLOG, FRL, KRL, PL2 etc). Development of tailor made IKBS has been made possible by the availability of purpose built Expert System building tools (eg SAVOIR, XI-PLUS, Knowledge Craft, Leonardo); these are essentially expert systems without a knowledge base.

3.3 LITERATURE OVERVIEW

For the items in the bibliography concerned with Expert Systems the following series of diagrams (Figures 3.3a-d) present a hierarchical classification according to subject matter:

• Expert Systems	ES	Figure 3.3a
• Intelligent Knowledge Based Systems	IKBS	Figure 3.3b
• Artificial Intelligence	AI	Figure 3.3c
• Applications	APP	Figure 3.3d

Those references providing a further literature review or bibliography are underlined. There is an approximate correlation between the classification and the sub-headings of this chapter; the diagrams may be used as an overview when reading the appropriate section.

FIGURE 3.3a Expert Systems - ES

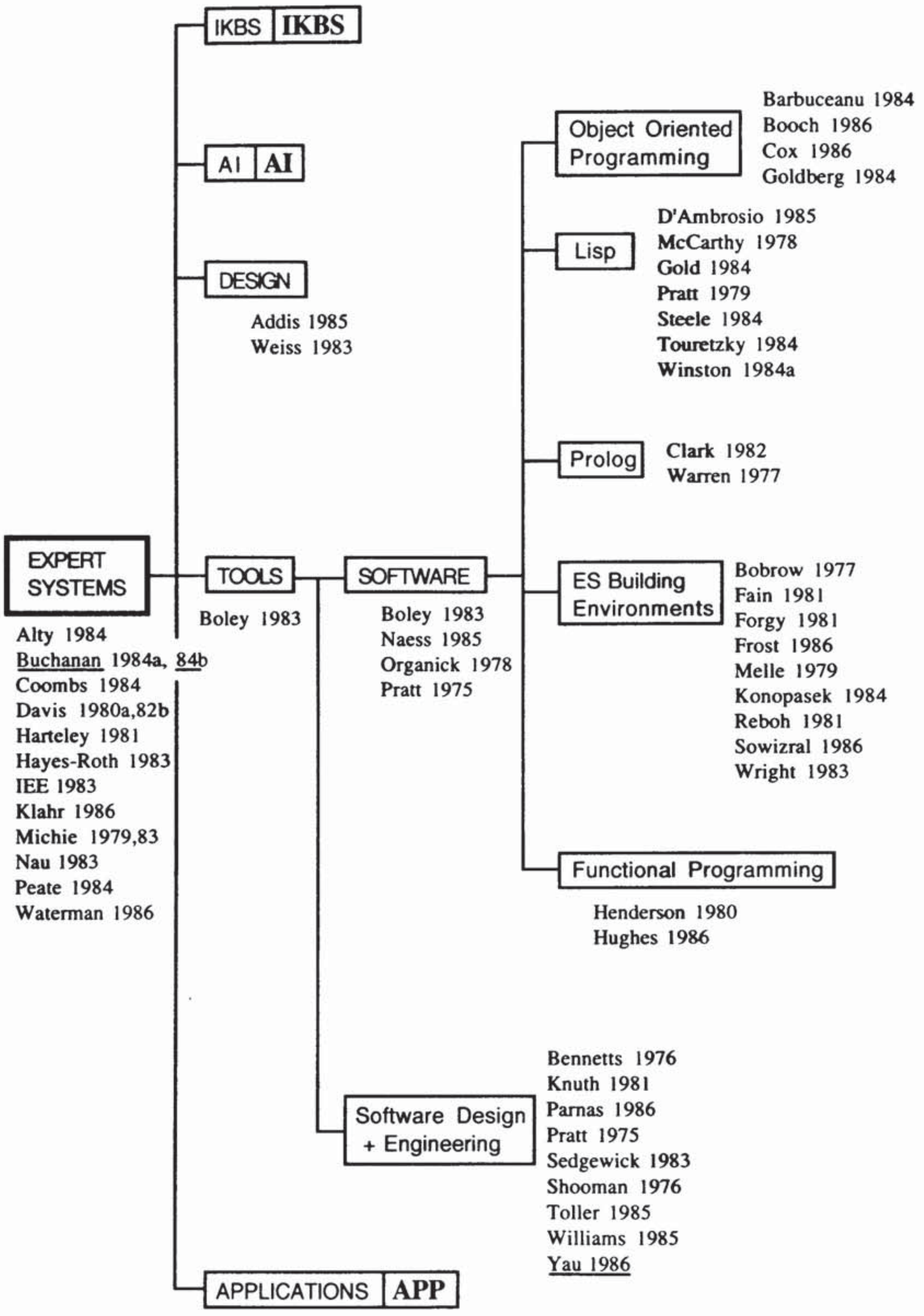


FIGURE 3.3b Intelligent Knowledge Base Systems - IKBS

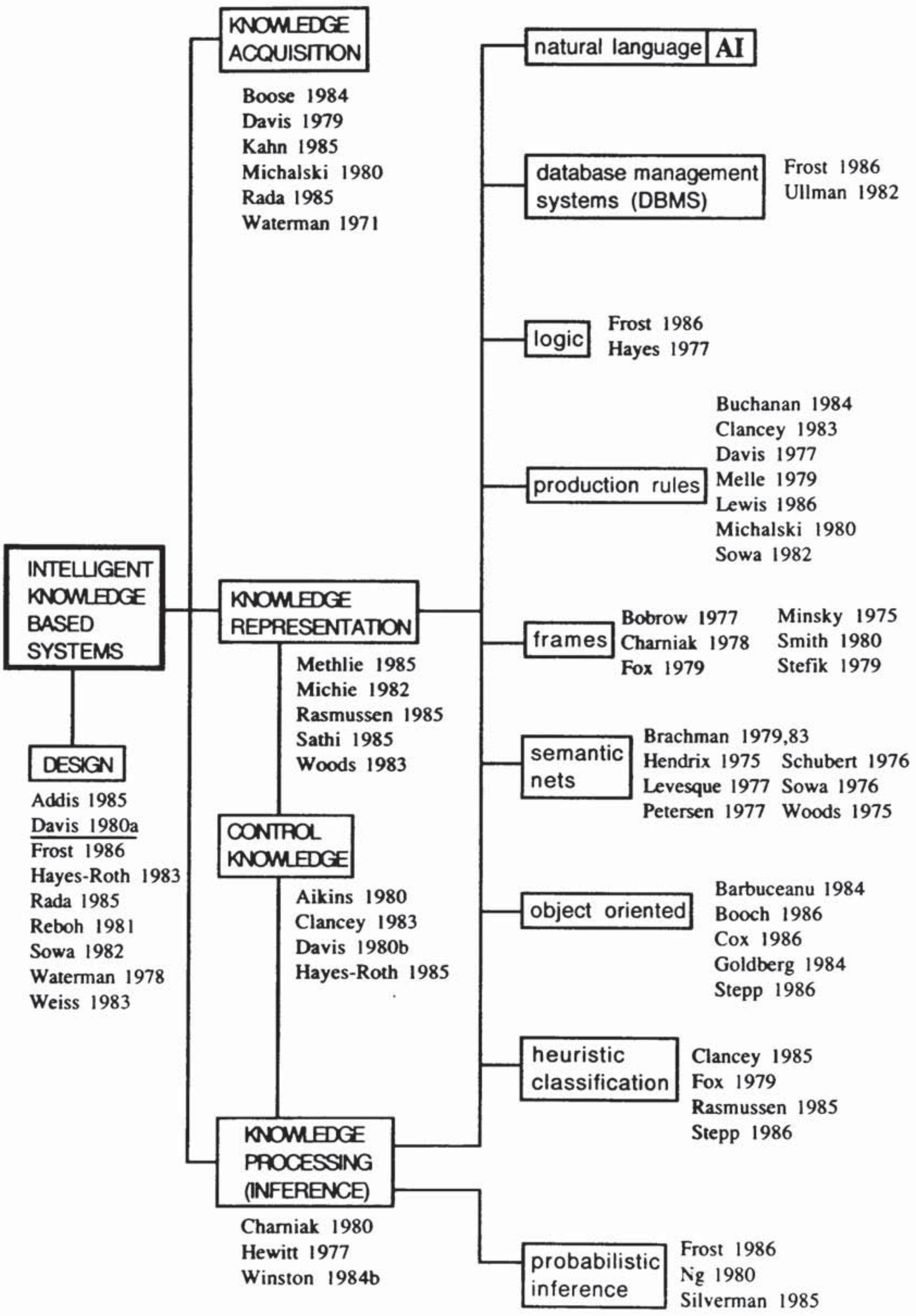


FIGURE 3.3c Artificial Intelligence - AI

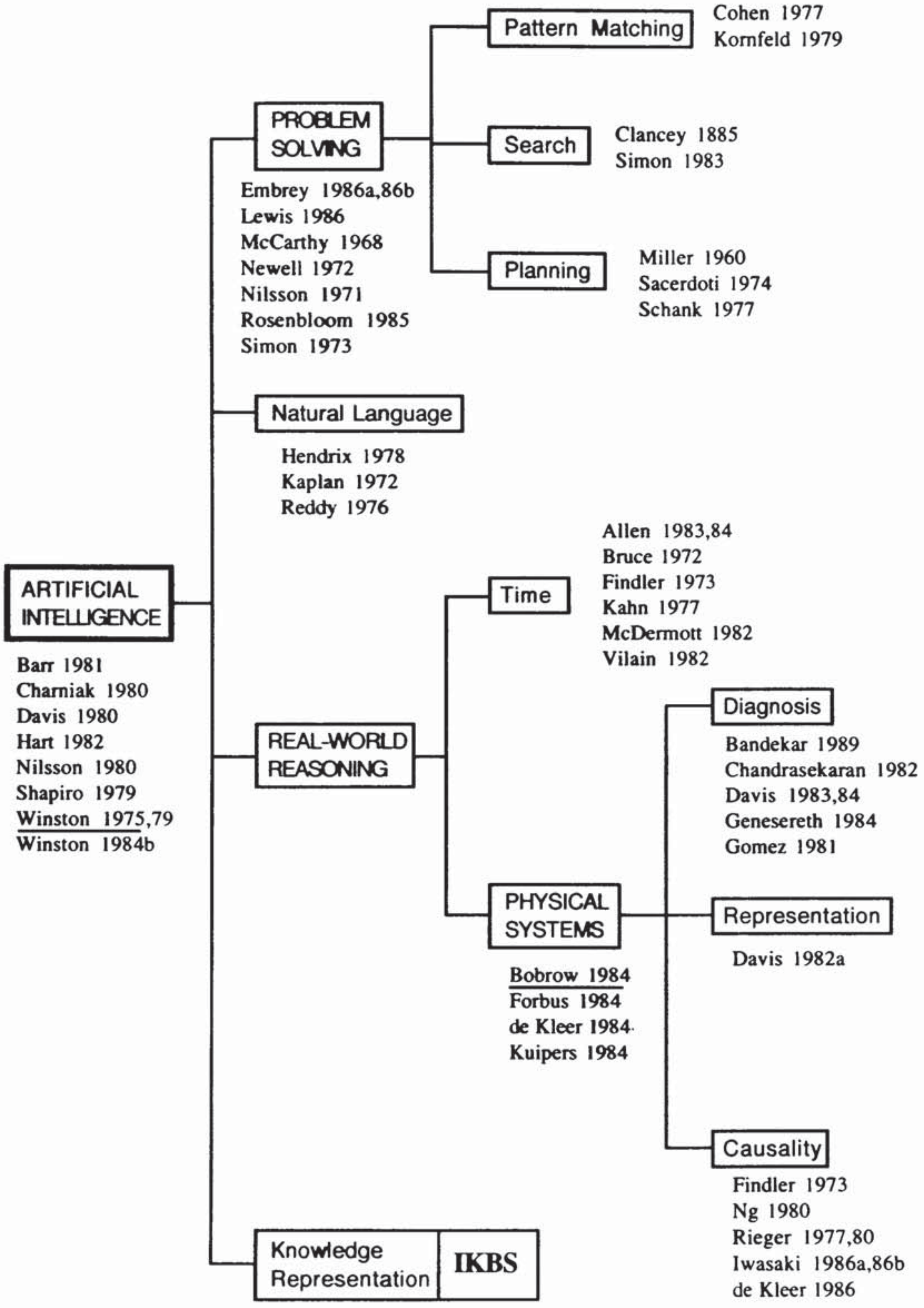
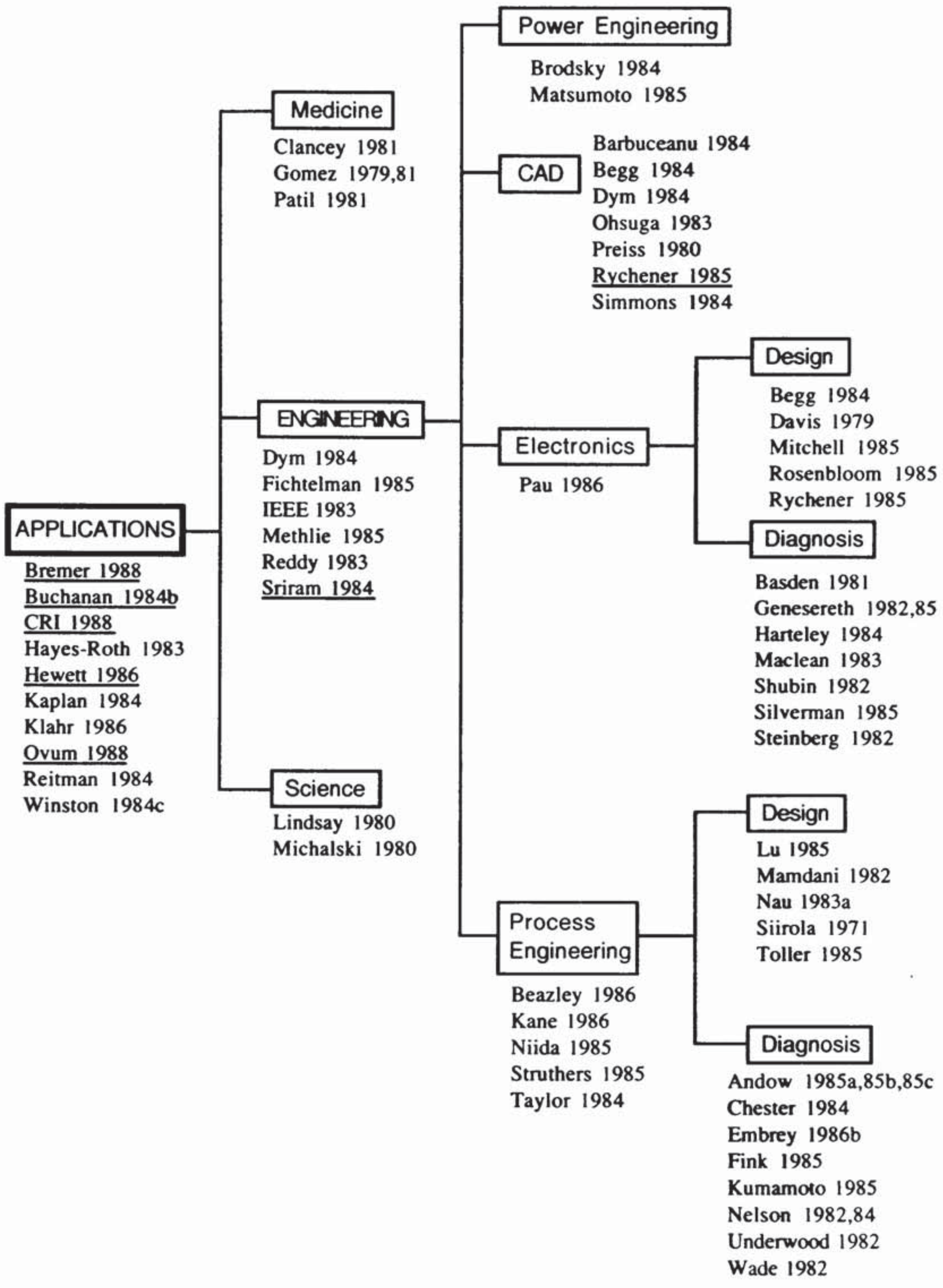


FIGURE 3.3d Applications-APP



3.4 COMPUTERS AND SOFTWARE

3.4.1 Computer Hardware

It is the nature of AI programs that they are large, requiring large memory and data storage capability, and they take a long time to run, necessitating processing power. For this reason they have been historically restricted to mainframes. Although useful work can be done on minicomputers and, more particularly, purpose designed workstations, personal desktop computers are still (even with their capability extended) considered to be severely lacking in power for this type of work. The reason why AI systems need this power in a computing system can perhaps be best illustrated using the example of a simple chess playing program. The program decides on a particular move by first considering every possible combination of board positions that can arise as a result of the next (say) five moves by each player. The number of different possible board positions is a very large number, for each additional move that is considered, the number of possible board positions increases by increasing orders of magnitude; this is known as combinatorial explosion. To generate these positions takes processing power and for storage, a lot of storage space will be used. The process generates a very large search space to be explored in selecting the best move, needing lots of processing power. In fact this exploration into the future of the game is done in a controlled fashion and is directed by a process known as heuristic search, exploring promising board positions further than non-promising ones. AI programs like this use symbolic languages where the data and program are large complex symbol structures. For example LISP symbol structures are stored using a cell/pointer system which is extremely greedy when it comes to memory space. This memory space has to be reclaimed every so often by a process known as garbage collection. AI programs also make use of recursive control structures, where a function may call itself. This process of self calling may be repeated to very many levels of depth, requiring large memory (stack) space to keep track of the return path. The author once inadvertently managed to completely tie up a VAX 11/750 minicomputer using recursion. Every time a recursion was initiated this created a new process which was allocated time on the machine. Very soon the program hogged 95% of the timesharing and all other processes ground to a halt.

AI Systems tend towards general models of intelligence and therefore often require more processing power than an IKBS performing a particular task in a restricted domain. Perhaps the best alternative for the latter kind of development is the personal workstation.

These modern machines have excellent processing and data storage capabilities and have the advantage over mainframes in that they do not operate on a time sharing basis. Exclusive use of the full power of the workstation allows truly interactive development in an excellent overall environment designed specifically for AI programming, in LISP or PROLOG, usually accompanied by more sophisticated ES building tools. The section on IKBS software which follows (3.4.2) discusses these environments.

3.4.2 **Knowledge Based Systems Software**

The software tools used for developing IKBS can be divided into two categories: Symbolic programming languages, of which the chief two are LISP and PROLOG, and expert system building environments, which free the knowledge engineer and the user from the mechanics of knowledge representation , inference and user interfacing, so that they can concentrate upon the problem to be solved.

LISP

The language LISP, List Processing, was invented by John McCarthy in 1958, though it was not in a recognizable computer form until a few years later. Historically there have been two major dialects of LISP, MACLISP and INTERLISP, each of which are in actuality far more than just a programming language but provide a complete programming environment. The unfortunate lack of standardization came to an end in the early 1980s when these two dialects, along with useful aspects of other minor variations, were brought together to form the now standard COMMON LISP. Versions of COMMON LISP are now available for all types of machine, from mainframe to PC, with only minor modifications required for transport of programs from one machine to another.

Programming in LISP is highly interactive. It is a functional language; to initiate a 'program' the user calls the highest level function of his 'program' which calls other functions according to its definition. This function calling, which may be recursive, continues until the primary LISP functions called primitives, with which LISP comes equipped, are reached. These primitives, operating on data structures as well as providing control, are evaluated, and the function call structure then 'unwinds'. Every function in LISP has arguments and returns a value, this is the 'purest' LISP way of passing information between functions, although LISP supports the full range of conventional programming capabilities. A LISP 'program' consists of a collection of function definitions.

Data Structures

LISP objects are either symbols or symbol structures. Primary objects are called ATOMS which are represented by simple symbols, eg a word, a number or the special values T and NIL, which represent the logical values true and false respectively.

Most data structures in LISP are lists. A list is defined as a set of zero or more elements each of which may be an atom or another list. This provides a very powerful mechanism for defining complex hierarchical data structures.

Variable assignation in LISP may be simple binding of an atom to a value, but this value may be any LISP object, data or function. Powerful mechanisms for data abstraction are provided, for example properties, eg the symbol MAN may have the properties bank-balance and address; each of these properties of the symbol MAN can have an appropriate value which can be assigned or retrieved by the access functions PUT or GET. One of the strengths of LISP is that data abstraction is very easy to implement by writing access functions for user defined data structures.

Functions

A function takes the form of a list. Prefix notation is used such that the first element of the list is the function name, and the rest of the list comprises the arguments that the function is to process, returning a value. Some examples of LISP functions are:

- List functions - operate to access, change or add to lists
- Predicates - to test a condition and return T or NIL

LISP comes with a set of primitives - built in functions. The user progresses by constructing a hierarchy of functions based on these primitives. This provides for functional abstraction (c.f. procedural abstraction). As data (lists) and functions (lists) take the same form syntactically, operations can be conducted on functions by functions. One advantage of this is in manipulating knowledge about procedures; functions may be created for a particular task and then evaluated. One example of the value of this is in manipulating knowledge about procedures.

<u>Control Structure</u>

LISP supports the full range of conventional control structures, eg iteration and conditional testing but as it is a functional language it is more natural to write recursively. This is an important concept, best illustrated by a simple example, a recursive definition of a factorial function:

```
To calculate Factorial(N)
          If N=1 then → Return the value 1
          Else → Return value given by evaluating N*Factorial(N-1)
```

<u>The LISP Environment</u>

To illustrate that working in LISP can be more than just using a programming language, it is instructive to look at the features of a LISP environment, as available on a purpose built AI workstation, such as those available from SUN, APOLLO or XEROX.

The first requirement is for an editor. LISP editors are written in LISP, and one such as EMACS provides for effortless editing of complex list structures (which are the programs). As functions are identical in syntax to data, powerful editing functions may be defined, possibly custom built by the user. Thus editing and evaluating (program execution) may be carried out side by side for application development.

LISP workstations support high resolution graphics, using window based systems similar to that used on Apple computers. The workstation may have the in-built capability to display graphs which represent, for example, the call structure of the user's functions or the network or tree representations of data structures, eg a hierarchical classification or a semantic net. This facility is also very useful for graphically displaying the propagation of inference in rule based systems.

IKBS tools may be embedded in the LISP language (or vice versa). This may be simply a ready built rule maintenance and inference structure or a fully fledged expert system shell, ie an ES without the knowledge. One example is the LOOPS system available in INTERLISP on the Xerox 1108. The value of the integrated development environment lies in the ability it gives the user to construct, manage, evaluate and use a knowledge base interactively .

Applications of LISP

LISP has been the primary tool for AI research and IKBS development, particularly in the US. Applications include:

- Implementing Rule Based Systems
- Symbolic Pattern Matching
- Object Oriented Programming
- Transition networks, eg Parse Trees defining natural language grammar
- Semantic Networks

PROLOG

PROLOG shares with LISP the characteristic of symbol manipulation, however, PROLOG stands for Programming in Logic. It is designed to answer questions (establish goals) using a knowledge base consisting of facts and rules. PROLOG works using predicate logic in the form of clauses representing either facts or rules which provide the knowledge of a particular domain (see 3.5.2 Predicate Logic). Establishing goals is done by a built-in inference mechanism which combines backward chaining with a simple backtracking strategy. PROLOG supports list structures but not with the same ease and facility of LISP. A large number of conventional programming abilities are also available (although the theory of computability states that any language which supports certain primitives can perform any computation task). Advanced environments are available for PROLOG programming, one example is the Knowledge Craft environment which uses a SUN workstation.

Whilst LISP predominates in the United States AI community, PROLOG has been more popular in the UK, at Edinburgh University for example, and in Japan, where it is said to feature strongly in the Japanese 5th generation project.

Other AI Languages

There are a number of other languages for programming in AI, many of which are derivatives of LISP and PROLOG:, for example LOGO, POPLOG, KRL, FRL, SMALLTALK etc. The more notable of these languages are aimed at a particular method of knowledge representation processing or programming. FRL was developed for manipulating Frame-Based knowledge representations (Section 3.5.2). SMALLTALK is an object-oriented programming environment.

3.5 KNOWLEDGE BASED SYSTEMS

3.5.1 **Introduction**

Although the term is still not one that has a widely accepted definition an Expert System (ES) may be defined as a computer system which could conceivably replace a human expert in a particular task. We would expect the ES to be characterised by similar attributes to the expert:

- Competence - high quality results in minimal time
- Knowledge - should be incorporated
- Works like an expert - rules of thumb rather than detailed theoretical knowledge
- Highly competent in a narrow specialised domain
- Gradual degradation of performance beyond the boundaries of the domain
- Able to offer explanations and justifications for the conclusions reached
- Able to supply a measure of confidence in the conclusions reached
- May make mistakes

In practice, many "Expert" systems do not approach the competence of a human expert but the majority of these systems are characterised by their emphasis on the explicit representation of Knowledge. They might be more accurately termed, then, Knowledge Based Systems (KBS). As may be seen from the definition of a KBS, systems covered by the terms ES and KBS may be considered to be in two sets which have elements in common.

Knowledge Based Systems

Frost [1986] gives the following definition of knowledge:

> "Knowledge is the symbolic representation of aspects of some named universe of discourse"

According to this definition a KBS will contain a knowledge base comprising such a symbolic representation and we would expect it to be written in a symbol processing language. This representation may be illustrated by considering a rule-based system, which is generally composed of facts about a domain and rules which relate those facts. Contrasting a KBS with a conventional Database (DB):

Knowledge Base - contains explicit facts and explicit rules (a lot of them)

Data Base - contains explicit facts and implicit rules (a very few)

This contrast also illustrates a difference between KBS and conventional computer programs, written in FORTRAN, PASCAL etc. In a conventional program the knowledge is stored implicitly in the form of the coded statements of the program; any alterations require a code change . In a KBS, knowledge is stored explicitly in the knowledge base as, for example, a set of rules or a graphical structure (eg network), and can be altered and analysed separately from the mechanism that processes the knowledge (called the inference mechanism) The knowledge of the system may therefore be considered to be modifiable rather than hard-wired. It can be managed independently of the procedures used upon it.

The term Intelligent KBS may also be used to define this type of system. This refers to a KBS which performs intelligently. A discussion of intelligence is more appropriate in the section on AI, 3.6

Overview of Section 3.5 - Knowledge Based Systems

An overview of the contents of this section on KBS, and the references quoted herein, may be gained from Figure 3.3b in the literature overview. KBS are best characterised by considering the different approaches by which they represent knowledge, as "all else follows". 3.5.2 considers knowledge representations whilst 3.5.3 considers how these knowledge bases are used in practice. This latter section on knowledge processing focuses on rule-based systems as these are the most common form of KBS and ES and may be used to illustrate more general principles.

A vital step in constructing a KBS is the acquisition of the knowledge to put into it. This is discussed in 3.5.4

3.5.2 Knowledge Representation

In developing a Knowledge Based System the choice of Knowledge Representation (KR) formalisms is the most important design decision. This choice determines, to a considerable extent, the options that are available for representing and reasoning with the domain knowledge. This section examines the major different types of knowledge representation which may be encountered. These are:

Natural Language	
Data Bases	
Formal Logics	Modular
Production Rules	
Semantic Nets	
Frames	Structural
Object Orientated Programming	
Scripts	

There are no clear boundaries between different KR schemes, and they share many factors in common. It is sometimes possible to translate directly from one representation into another, eg from frames to predicate logic. Methods of processing one type of knowledge may be useful in processing others, for example search and conflict resolution strategies of propositional logic are often directly applicable to rule based systems.

Each representation has its strengths and weaknesses and each is appropriate for a particular task, ie a particular means of knowledge processing. For example, Frames may be used for representing objects and production rules for reasoning about them.

The different KR schemes may be divided into four groups:

Natural Language - not a formal KR method therefore of limited use

DataBases - storage of large amounts of simple data with little semantic content

Modular KR - representing "bites" of context free knowledge

Structured KR - reflecting the structure of the domain represented

Natural Language

Natural language possesses great expressive power, and is readily understood and used by the vast majority of people. So why not use natural language for knowledge representation in a KBS?

Any language may be defined by its syntax and its semantics:

Syntax - a set of rules which describe the way in which legal expressions may be formed in the language, ie defines the construction of sentences in natural language (the concept of a syntax is similar to that of a grammar)

Semantics - defines the attribution of meaning to legal expressions in the language

In order to use a language of representation and inference in a KBS, this language must have a rigidly defined syntax and semantics; it is then referred to as a formal language. Computers cannot work with non-formalised data.

Natural Language (in this sense, for example, general English) cannot be used for KR as no one has yet succeeded in formally defining its syntax and semantics. The understanding of natural language is one of the core problems investigated in AI work (section 3.6). It is, however, possible to define a syntax (a grammar) which defines a subset of natural language. From such a grammar it is possible to construct semantic interpretation mechanisms for translation of an expression in a language into another language or other knowledge formalism. However, grammar of this type will only define a single sentence at a time; this sentence is not considered in the context of the body of text of which it forms a part, or the belief system underlying the semantic interpretation of the sentence. For this reason it is described as a context-free grammar.

Consider the following passage:

"My car has broken down. It is the one by the lamp-post with the flat tyre"

The second sentence illustrates:

Context Dependency: "it" refers to "my car" which is described previously.

Global Interpretation: One has to know that a car may have a flat tyre but that a lamp-post cannot.

These are major problems with natural language, and even of subsets with formal syntax. Semantic interpretation of legal expressions is context dependent requiring:

(1) Construction of a model of the universe according to previous and following sentences.

(2) Availability of a belief system or general knowledge base of the universe as a whole.

Although progress has been made in the development of context-dependent natural language understanding, the second problem, defining general knowledge of the universe, is obviously intractable. This implies that for the foreseeable future, natural language understanding will be restricted to knowledge of narrow domains using a sub-set language. Pseudo-natural languages are of considerable value in the human-computer interface, and many application programs make good use of them, for example database query languages. In KBS, a grammar based system may be used to translate internal knowledge representations into a pseudo-natural language so that they may be more readily understood by the user.

Databases

Data base technology is well developed in the storage of information, its retrieval, modification , sorting etc. However this type of knowledge consists of a large number of simple facts with very few rules for representing knowledge about these facts. These rules are implicit in the Database Management System (DBMS) which manages the data. It is possible to represent entities and relationships between those entities but not much else, for example there is no provision for explicit inference knowledge or for representing alternatives, implication or negation. However, large amounts of data are used in KBS and it is useful to be aware of the various DBMS techniques for handling this data. These include record structure types, relational and binary relational DBMSs.

Formal Logic

There are many different types of logic including classical propositional logic, first order predicate, sorted first order predicate, non-monotonic, modal, temporal and intensional, each with their own uses in formal reasoning and theorem proving. All types of formal logic are languages of representation with two characteristics:

1) A rigidly defined syntax for defining 'legal' expressions, which may be expressed graphically, or in text format. A good example of a syntax definition is that of the language PASCAL.
2) A rigidly defined method of manipulating expressions in order to facilitate reasoning, for example *modus ponens*, which states that: from A and A ->B we can infer B.

It is useful to consider two basic forms of logic as they have an important bearing on the structure and manipulation of other knowledge representation formalisms. These are propositional logic and predicate logic.

Propositional Logic

Expressions in propositional logic are constructed from atomic formulas (assertions which have truth values) and the logical connectives:

$\wedge$	=	AND
$\vee$	=	OR
$\neg$	=	NOT
$\rightarrow$	=	IMPLIES

It is important to note that the atomic formulae in propositional logic are semantically indivisible, consider:

(Calvin has a tail) $\wedge$ (Calvin is furry) $\rightarrow$ (Calvin is a cat)

Each of the bracketed terms is a self contained statement which cannot be divided. It is not possible, from this statement, to generalise to the statement that all furry things with tails are cats. Another expression would have to be defined in order to show that Hobbes is a cat for the same reasons as Calvin. This is the chief disadvantage of propositional logic: there is no facility for expressing specific knowledge about entities, general knowledge of classes of entities (entity-set membership) or relations between entities.

The expression: $A \wedge B \rightarrow C$ may be rewritten as: If (A and B) Then C. This is the representation used in production rule based systems, which use the modus ponens manipulation rule to support reasoning. In a rule base, A and B may be complex terms in themselves and are not necessarily semantically indivisible. In considering rule representation in the section which follows it is important to remember that some methods of inference, such as forward and backward chaining, and search, such as breadth first, depth first, backtracking and more complex search strategies, were originally developed to handle logical reasoning.

Predicate Logic

Predicate logic essentially overcomes the semantic indivisibility of the atomic formulas of propositional logic through their replacement by predicates, eg:

1) cat(Calvin)

2) owner(Calvin,Sue)

3) $\forall x[[\text{furry}(x) \wedge \text{has-tail}(x)] \rightarrow \text{cat}(x)]$

4) $\forall x \exists y[\text{pet}(x) \rightarrow \text{owner}(x,y)]$

1) Denotes that the entity Calvin is a member of the entity-set 'cat'.

2) Defines a relation between two entities

3) For all x, if x is furry and x has a tail then x is a cat. This illustrates two principles:

   Generalization - any entity may be considered for x

   Implication - from one formula we can infer another

4) For all x there exists a y such that if x is a pet then y is the owner of x. This illustrates the implication of the existence of new entities.

The logical connectives for predicate logic are those for propositional logic plus $\forall$ - meaning "for all", and $\exists$ meaning "there exists".

Predicate logic has a relevance to more sophisticated knowledge representations. The knowledge represented by simple frames or semantic nets may be directly translated to predicate logic. PROLOG uses this logic in the form of Horn clauses [Robinson 1987].

Both propositional and predicate logic are useful representations when it comes to explaining the behaviour of a particular application of KBS, as here for fault analysis.

Production Rules

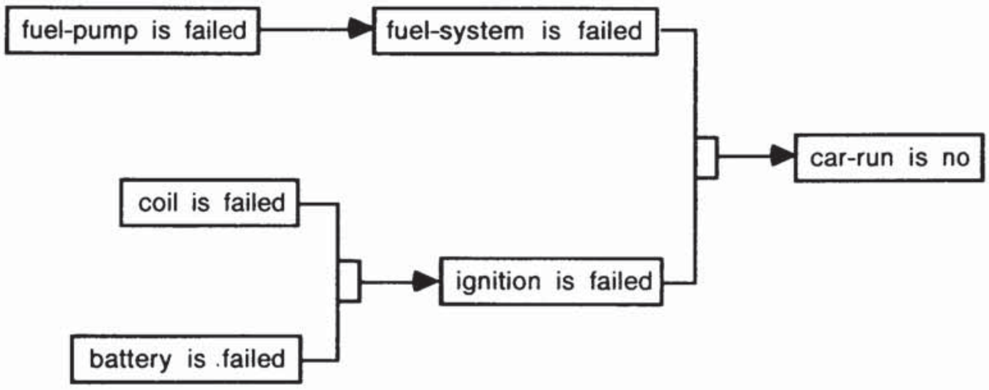
A production rule defines a small self contained chunk of knowledge about a particular subject. This knowledge is in the form of implications which may be used to generate new facts based on existing ones. Consider the rules in Figure 3.5.2a.

FIGURE 3.5.2a Production Rules and their Representation

IF (fuel system is failed) **OR** (ignition is failed) **THEN** (car-run is no)

IF (fuel-pump is failed) **THEN** (fuel-system is failed)

IF (battery is failed) **OR** (coil is failed) **THEN** (ignition is failed)



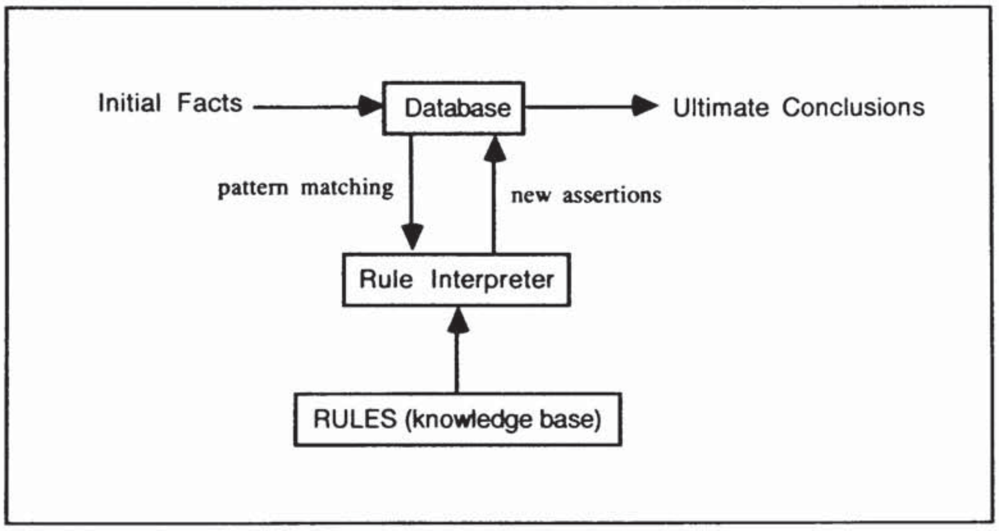
This tree representation of the relationships between concepts may be constructed first, and the equivalent rules derived from it.

The rules are similar in form to implication statements in propositional logic and the same logical connectives apply: AND, OR, NOT, IMPLIES. However the propositions in the rules (bracketed terms) need not be atomic, they may represent anything the user wishes, here attributes of objects. A more general form of production rule is IF C THEN

A. C is a condition to be met, usually a pattern of facts (using operators AND and OR) to be matched to existing data. If C is satisfied, the rule "fires". A is an action to be taken if the rule fires, this may involve adding new (implied) facts to the database of facts or the undertaking of a procedural task, such as querying the user.

The structure of a simple rule based system may be that shown in Figure 3.5.2b. The mechanism which applies the rules to the facts existing in the database is the rule interpreter, or, more generally, an inference engine. The process is one of inferring goals (desired conclusions) from data (initial assertions)

FIGURE 3.5.2b Architecture of a Simple Rule Based System



Reasoning With Rules

There are two main ways of reasoning with production rules, these are illustrated with respect to the simple forms of rule representation above:

1) Forward Chaining

i. A set of assertions is provided in the database.

ii. The conditionals of all rules are matched against the assertions.

iii. For each rule that matches, the action of the rule is executed, adding new assertions.

iv. ii and iii are repeated until the goal assertion appears in the database or no rules can be matched.

Forward chaining is sometimes referred to as 'data driven search', as reasoning is forward, from data to goals by chaining rules together transitively. This chaining is illustrated in the left-to-right tree structure of Figure 3.5.2a.

2) Backward Chaining

i. A goal is selected. This goal is an assertion to be established as true or false (eg car-run is no)

ii. The action parts of all rules are matched against the goals in the rule base

iii. For each rule that matches, an attempt is made to establish the truth of the conditional statements of the rule. These may be initial facts - can query the user - or conclusions of other rules - these are sub-goals to be established and are added to the database.

iv. ii and iii are repeated until the goal is established (by finding sufficient true primary assertions) or proven false.

Backward chaining is sometimes referred to as goal-directed search. In practice an inference engine may combine both forward and backward chaining.

At any one iteration of the matching of rules to assertions in the database there may be more than one matching rule. The question then arises as to which rule to apply first. This is known as conflict resolution. There are a number of strategies, from selecting the first rule that comes along, to making a best guess based on pseudo-probabilistic data associated with each rule.

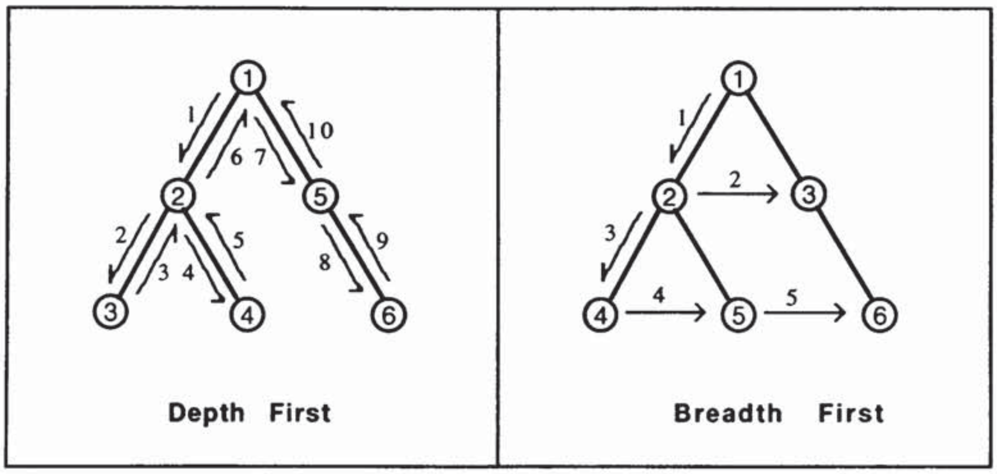
Search

As shown in the tree structure in Figure 3.5.2a the rules define a state space for the universe of discourse. This is referred to as the search space and in systems of any complexity which may contain many generalised rules, the search space may be very large indeed (cf chess example).

Consider the backward chaining approach above. In order to establish the goal the tree is searched from the goal to the primary assertions but how is this searching to take place? There are two approaches, illustrated in Figure 3.5.2c:

- Depth first search: each node at a given level is investigated fully before the consideration of the next node at the same level.
- Breadth first search: all nodes at a given level are investigated before moving down to the next level.

FIGURE 3.5.2c Search Mechanisms



These strategies are both exhaustive and in the case where the search space is very large this is impractical. A more ad hoc strategy may be more appropriate, using rules of thumb or general, experiential knowledge to guide the search; this is known as heuristic search [Clancey 1985].

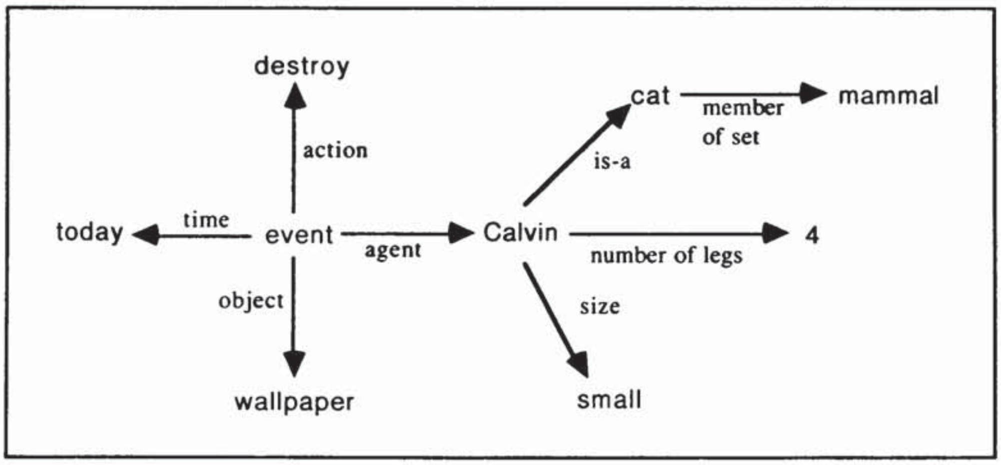
Structural Knowledge Representation Formalisms

Production rules allow us to represent information about a domain but they do not permit the structuring of this knowledge to reflect the way in which the domain knowledge is structured. Such topological KR schemes include semantic nets, frames, conceptual dependency structures and scripts.

Semantic Nets

A semantic net is a graphical representation describing concepts or entities and the relationships between them. An example is given in Figure 3.5.2d.

FIGURE 3.5.2d A Simple Semantic Net



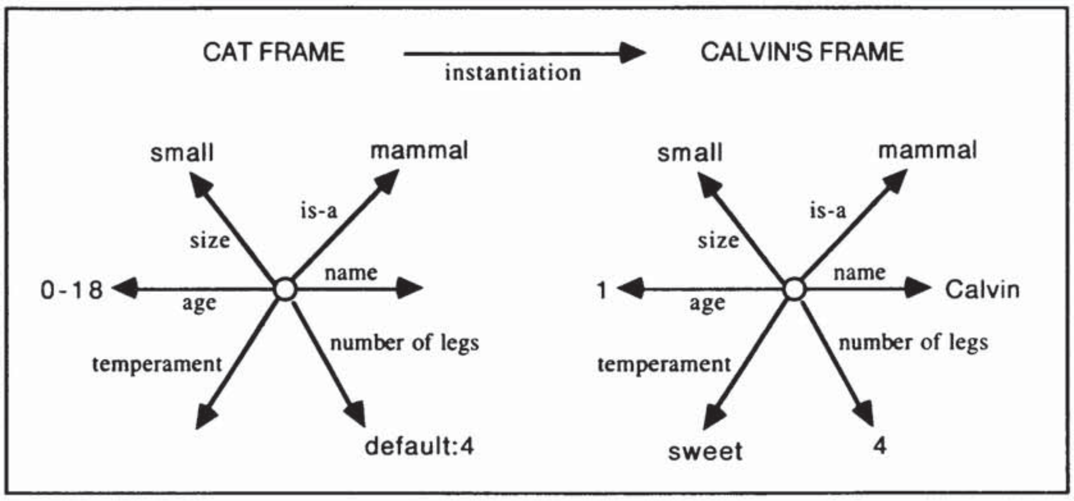
Nodes are used to represent entities in the net and binary relations are shown by named arcs.

The basic semantic net representation has been developed in various ways, eg extended semantic nets or partitioned semantic nets.

Frames

A frame consists of a collection of assertions about a particular entity type. These assertions, or attributes of the entity are held together as shown in Figure 3.5.2e. Each attribute consists of a named slot which may be filled by a value.

FIGURE 3.5.2e Frames



Of particular significance is the notion of a stereotype frame. This is, in essence, an empty frame characterizing a given type of entity. When the slots of the frame are filled in order to describe a particular entity , the frame is said to be instantiated.

The slot value of a particular frame may be initiated by the user or may have a **default** value, or choice of values specified in the stereotype frame. Where the slot value is predefined as the stereotype (eg small size for cat) this is referred to as a generic value, common to all instances of the frame. This value may be a simple attribute of the entity or it may indicate a relationship to another entity; in this case the value is a pointer to another frame. Such a relational slot may indicate set membership by pointing to a frame representing the set of all such entities. A slot may also have a procedure associated with it - this is called a demon, as it is automatically activated whenever a certain condition pertains to the slot, ie it is instantiated, changed or accessed. For example, the demon may be intended to check the slot value to avoid conflict with other similar entities.

One commonly used attribute in frame systems is the 'is-a' relationship pointing to another frame. If we say that <x is-a y> , then x is a specialization of the more general object y. For example, w and x may be the frame for cat and dog respectively whereas y may be the frame for a mammal. Here both <w is-a y> and <x is-a y> , both cats and dogs belong to the general class of mammals. The formulation of an is-a hierarchy allows for the inheritance of properties by the **child** frames from the **parent** frames, where <child is-a parent>. In our example both the cat and dog frames will inherit slots from the mammal frame, for example the property warm-blooded, has lungs etc.

Matching

If an unknown object is encountered it may be matched against the stereotype frame to see if it conforms to an existing object type. This may lead to either a positive match or a best match, in each case the next aim is to fill any needed but empty slot values in the object by a kind of guesswork using the slot values of existing instantiations of that stereotype.

Inference

There are various means of inference available in frame based systems:

- When a match is made we may infer that the object belongs to the general class of the matching stereotypes, allowing attribution of generic values.
- The is-a slot defines generic inheritance of properties
- Analogies may be drawn between objects, eg a hamster is like a very small cat but does not live for very long. Some rules for hamsters may be applicable to cats.

Object Orientated Programming

The notion of object orientated programming arose out of the representation of objects as frames. The expressive power of object orientated programs relies on exploitation of the demon properties to allow for objects to communicate by **message passing**. Messages are procedural instructions which may be sent by the user to an object, from object to object or object to user.

In more advanced object orientated programming each object has a message interpreter which tells it how to decode a message it receives and then to put it into action. Action may involve retrieval or changing of object attributes, sending messages to other objects, or engaging in defined tasks. This raises the possibility of creating a real-world simulation by defining objects to represent world entities and their behaviour with respect to each other and to external events.

Scripts

A script is a structured set of assertions, actions and procedures representing a particular sequence of events. It is based on the idea of conceptual dependencies, originated by Schank [Schank 1977]. Conceptual dependency structures are essentially sophisticated semantic nets involving a complete ontology of entities, actions, conceptual tenses and dependency between concepts. Scripts are used for reasoning about procedures. One example of their use in inference is that if a script is begun we can infer that the rest of the script will follow.

3.5.3 **Knowledge Processing**

The previous section considered various ways of representing knowledge and included a description of the ways in which this knowledge can be used to reason about a domain. In this section advanced techniques of Knowledge Processing (KP) are discussed with specific reference to rule based systems, as these are by far the most common Expert Systems. The topics discussed here are: reasoning with uncertainty, the use of partitioned knowledge bases and the representation of control knowledge.

Reasoning with Uncertainty

Expert Systems reason about a subject at a high level of abstraction, using rules of thumb or more generally, heuristics. Heuristic knowledge is that of experience and may not be deterministic; it may consist of a belief in a certain implication, with a certain strength behind that belief. For example, consider the rules:

1. IF pound falls THEN interest rates rise
2. IF interest rate rises THEN mortgage rates rise

The first rule is not necessarily true, an expert would express a certain degree of belief, perhaps a probability that it would be the case. The second rule however is almost certain. Another source of uncertainly arises where data may be incomplete, unreliable or vague, eg is tall, feels cold.

There are two main types of uncertainty reasoning methods, probabilistic inference and fuzzy logic.

Probabilistic Inference

These methods are based on Bayes Theorem for conditional probability. In a rule based system this involves assigning probabilities for:

- initial assertions or data
- probability of implications ie $X \rightarrow Y$ with a probability of P
- *a priori* probability of the outcome, ie the prior probability that Y is true, independent of any evidence

Bayes Theorem provides a means of computing the probability that a hypothesis (H) is true based on evidence (E) related to that hypothesis:

$$p(H:E) = \frac{p(E:H)p(H)}{\sum_{1}^{n} p(E:H_i)p(H_i)}$$

$p(H:E)$ =	probability of H being true given that evidence E has been observed
$p(E:H)$ =	probability that E will be observed given that the hypothesis H is true
$p(H)$ =	*a priori* probability of the truths of the hypothesis, independent of any evidence E, eg the prior probability of throwing a 6 with a die is 1/6
n =	the number of possible exclusive and exhaustive hypotheses

Bayes Theorem was used by the author in a simple expert system for the identification of wild mushrooms. In this case each Hypothesis is that the mushroom to be identified - the specimen - is a given species, and each piece of evidence is a characteristic that the specimen may exhibit.

In practice the way in which uncertainty is handled may not be based on true probabilities, but is rather pseudo-probabilistic, eg "probabilities" are not strictly calculated according to Bayes Theorem. This is not so important as long as it works, after all building expert systems is often a pragmatic process. A good example is MYCIN - which made use of ad hoc certainty factors and calculations.

Fuzzy Concepts

An alternative approach to reasoning with uncertainty is based on the degree of belief associated with assertions or implication, in the form of certainty factors. Consider the rule IF X THEN Y with certainty C: This certainty factor (CF) commonly ranges from -1 to +1 , indicating, respectively, a total lack of confidence in the implication or a 100% belief in it.

The initial assertions on which the rules are based may also have an associated certainty factor. These different factors may be combined arithmetically to determine a certainty for the implied assertions (Y in this case)

Fuzzy logic also includes the concept of associating a term with a range of possible values. For example the terms hot, warm, mild, cool and cold each refer to a range of temperatures.

Control Knowledge

In large knowledge bases containing many rules it is extremely time-consuming to consider all the rules at all times when only a subset of them is required for the task at hand. This kind of thinking leads to rules about rules, these are called meta-rules and may be used to store control knowledge as a knowledge base, with all the advantages that brings. For example, a meta-rule might be:

IF problem is (engine will not run)

THEN use ignition system and fuel system rules to establish (engine will not run)

Partitioned Knowledge Bases

Blackboard Architectures

It may be the case that a problem is too complex for one expert system to solve by itself. It may be advantageous to partition a knowledge base into several constituents, each with a particular bearing on a problem, for example we might have one KBS that decides on the possible occurrence of faults in a system and another that determines how these faults may propagate. In these cases there is a need for communication between knowledge bases. A blackboard is a separate data structure which may be added to or modified by a KBS. The individual KBS may observe the blackboard waiting for the appearance of information which is in its domain and then act on that information to make changes of its own to the blackboard.

3.5.4 **Knowledge Acquisition**

[Davis 1979, Kahn 1985, Rada 1985, Waterman 1981]

There are two general approaches to the acquisition of knowledge for inclusion in Expert Systems. These are

1) Direct Acquisition - being told.

2) Learning - acquiring or selecting the rules describing the domain by exposure to examples.

1) Direct Acquisition [Boose 1984, Hayes-Roth 1983]

The direct acquisition of knowledge requires the elicitation of that knowledge from an expert, and the encoding of the knowledge in a representation suitable for a KBS. An expert, or alternative source of knowledge, is therefore required. This is the conventional way of constructing expert systems, described in section 4.4.

2) Automated Learning [Michalski 1980, Rada 1985]

The subject of machine learning is one of the main problems tackled in AI research but here the aim is to develop general models of learning behaviour. In ES development we are interested in acquiring specific knowledge about a domain and we may well have concrete examples of the tasks which we want the system to perform. To this end, automated learning, learning by induction from examples, takes the following form:

1 The data for an example problem is provided.

2 The ES attempts to solve the problem.

3 The solution of the system is compared with the example solution.

4 If the solution is correct, reinforcement of the reasoning used is carried out. If the solution is incorrect, the reasoning used is reduced in importance.

Thus a simple learning program may be provided with a subset of all the rules possible for a given domain (mapping of data to solution) and may weight those rules using a probabilistic or certainty factor principle in order to learn. The general principle behind learning of this nature is simple modification of implication through feedback.

Presenting an ES with data and solutions gives no clue as to how the solution is to be reached, unless single step inference is used. It may be advantageous to precede the learning process by the direct acquisition of knowledge identifying key concepts and interrelationships. This identifies "stepping stones" between the data and solution thereby "fleshing out" the mapping of rules from one to the other and defining sets of rules which may be applicable.

More recently there has been considerable interest in the use of neural networks which gain competence in a task by a process of pure learning from example and direct modification through feedback. This is, however, outside the scope of this work.

3.6 ARTIFICIAL INTELLIGENCE

3.6.1 Introduction

Artificial Intelligence (AI) is based on the assumption that the best way to understand intelligence is to attempt to reproduce it. More specifically, the aim is to artificially produce evidence of intelligent behaviour, whatever that is. This reproduction is, by default, conducted by computer. It is interesting to contrast AI with experimental psychology. An analogy may be drawn with physical systems. There are two basic approaches to understanding a physical system, the empirical and analytic: either to conduct experiments on it, or to develop analytical models of it, in the physical case mathematical models for numerical simulation. The two different approaches of AI and experimental psychology were always somewhat at odds until the late 1970s when the two fields were brought together, along with linguistics, formal logic and philosophical considerations to form the new discipline of cognitive science. An understanding of cognitive science is useful in Expert Systems work as this science attempts to answer questions concerning how people perceive and reason about the universe, for example, what models of the universe do people have in their minds and how does an expert perform a particular task.

AI research has focused on the following principal problem areas:

- Natural Language - recognition and synthesis
- Thinking and Reasoning - problem solving, theorem proving, planning
- Learning
- Representation of knowledge
- Computer Vision
- Real world reasoning

A lot of AI work is somewhat tangential to building IKBS. Aspects such as computer vision are clearly not applicable but more generally there is a strong difference of emphasis in the two subjects. In KBS emphasis is on quantity, quality and diversity of explicit knowledge used to solve a particular set of problems in the real world. In contrast

AI attempts to produce general models of particular aspects of intelligence. This often involves developing a neat and complete symbolic computer program to solve a set of problems in a very simple artificial world. This is aimed at demonstrating the success of a particular theory of some aspect of intelligence, reasoning method or view of the world. In other words these models are widely applicable but only support reasoning in a very simple domain. However it must not be forgotten that practically all developments in Expert Systems owe their existence to AI research. AI concepts, theories, taxonomies, ontologies, models, representation and reasoning techniques provide new and extend existing IKBS methods.

This section focuses on particular areas of AI research principally representing and reasoning about physical systems in the real world. As it turns out, this appears to be a small part of AI but is gaining in significance as a result of the upsurge in ES/IKBS.

3.6.2 **Reasoning About Physical Systems**

Reasoning about physical systems in AI uses symbolic rather than numerical models and may be referred to as **qualitative reasoning** (QR) A volume of the journal Artificial Intelligence was devoted to the discussion of qualitative reasoning about physical systems in 1984 [Bobrow 1984]. and this section discusses briefly these approaches with expansion on those topics relevant to the problem at hand.

Some general points about qualitative reasoning emerge from a consideration of the literature.

1) The behavioural description of the system must be **compositional**; the behaviour of the system is derivable from the structure of the system, ie the components, component behaviour models and the interconnections between components.

2) <u>Causality</u>

Differential equations which describe a system only impose **constraints** on the dynamics of system variables and do not describe the **causality** (the flow of cause-effect) within a system. In qualitative models causality should be seen to propagate through the system from component to component via connections.

3) <u>Discrete vs continuous variables</u>

In mathematical models variables are continuous, ie they may have any value within a given range. In qualitative reasoning, variables have discrete values (eg high, normal low) which must map onto a range of equivalent continuity. This implies that QR is most readily applicable in the case of discrete systems such as digital electronics rather than continuous systems such as process plant. For this reason a large amount of work has been carried out on digital systems whilst successful modelling of continuous systems has been confined to a few simple examples.

4) <u>Quantity Space</u>

The term quantity space refers to the different values that a variable may take. In AI QR this is generally restricted to a simple set, eg {-,0,+}, or even {0,1} for digital systems. Obviously this is insufficient for chemical processing systems, consider the HAZOP Guide Word quantity space, defined in Table 2.4.3a where the variable "flow" has the following quantity space: {no, more, less, as-well-as, part-of, reverse, other-than}. The concepts embodied in these guide words are too complex for any simple homogeneous quantity space. This is a serious problem with the very neat but restricted QR systems which have been proposed.

5) <u>Function</u>

It may be useful to reason about the function of an object as distinct from its structure and behaviour. [Rasmussen 1979a]. For example, the function of a relief valve is to prevent damage to the vessel which it is protecting; we may reason that failure of the valve implies the existence of a hazard to the vessel.

6) Primitives

Qualitative reasoning about causality in physical systems may be characterised by the primitive elements of the representation from which causality is determined. These are either variable centred or device centred representations.

i) State Variable Centred Representation [Kuipers 1984, Bandekar 1989]

In this paradigm the ontological primitives are state variables with qualitative constraints describing their interdependence. These models may be depicted graphically as a network of nodes corresponding to state variables, with bidirectional arcs between the nodes indicating the permitted qualitative values they may have with respect to one another. These variables and their interactions may be decided upon by collecting the differential equations that describe a device and replacing them with their qualitative equivalents in a manner similar to that used by Andow [Shafaghi 1984] and Taylor [Taylor 1982] in developing models of process devices.

This methodology bears a striking resemblance to that of Lapp+Powers [Lapp 1979] who use variable digraphs to represent interactions between process variables. In variable digraphs, however, directed edges are used to indicate the flow of causality from one variable to another whereas state variable networks represent **constraints** between variables. Causality is derived from the variable network by a process of constraint propagation or techniques known as **causal ordering** [Kleer 1986]. Alternatively causal ordering may involve references to the boundary variables of the system and the topology (digital schematic, P&ID) of the system.

There are three basic levels of causal dependency that may be defined between variables:

1) Pure constraints on variable values. Individual variable values are tied together such that the existence of one variable value implies the existence of others, and vice-versa. The constraints may be defined as sets of consistent variable values. Note that there is no information regarding the direction of causal propagation.

EG (A:low, B:high) (A:high, B:low)

2) Bidirected causality. A change in one direction of a variable value implies a change in a certain direction in another, and vice-versa. Again, the direction of causal propagation is not defined.

EG (A:increase, B:decrease). If A increases, B decreases. If B decreases, A increases.

3) Directed causality. In a one-way causal link the direction of change in one variable is directly responsible for the direction of change in another. This is the method used in the variable digraphs of Lapp and Powers. Note that causal propagation is defined.

EG If A increases, this causes B to decrease.

In addition to these methods, it is clearly possible to directly describe causal links between events defined as variable values. EG If {A high} occurs, this causes {B low}. This is not, though, a variable centred representation.

The state variable network approach suffers from the same problem as the variable digraph method in that synthesis of the graph involves almost as much effort as direct synthesis of the causal analysis. The graph must first be synthesised from the structure of the system. This has not yet been demonstrated for physical systems of any complexity.

ii) Device Centred Representation

In this methodology the system is described in two ways:

Topology - The identity of the system components and their interconnections.

Component behaviour Models - representing input/output information.

Again, this approach is almost universally confined to modelling causality in digital systems. Digital devices are easy to identify as they are discrete in both topology (at both an ontologically primitive and abstract level) and behaviour (defined by formal logic). Although chemical processing systems do contain equipment which may be regarded as discrete devices, eg valves, transducers, with clearly defined i/o and failure modes, a device based method is somewhat stretched when it comes to handling complex entities such as streams, reactions, VLE, flow/pressure in pipes, mass transfer etc.

A very important consideration in device model selection is that the model must not contain embedded assumptions which make its behaviour dependent on the context in which it appears. This context dependency is an important consideration in process systems where flows of material and energy may be misdirected.

<u>Conclusion</u>

The chief disadvantage of Qualitative Reasoning as a means to solve problems of causality in physical systems is that it essentially attempts to construct direct symbolic models of the world. There are two problems with this:

1) To be comprehensive, symbolic models must approach the complexity of mathematical models, indeed the symbolic models are derived from differential equations developed in the conventional way. QR must ultimately be rejected as a solution for the same reason that full mathematical models of chemical process systems are not generally produced - it is too expensive and time-consuming. In the development of rules for generic unit models though, prior consideration of differential equations may prove useful.

2) This type of QR is restricted to reasoning about physical systems in very basic mechanical terms; there is no diversity of knowledge and certainly no expertise - that higher level or heuristic knowledge that characterises the way in which a human expert would approach the problem.

Although QR as a methodology is not directly applicable to the problem at hand, the concepts introduced, defined and classified and the general aspects of the models of representation and causality discussed prove very useful. Particular concepts have been directly applied in the suggested expert system design:

- Compositional behavioural description.

- Discrete variable quantity space.
- Primitives - device and variable centred.
- Representation and propagation of causality.
- Distinction between structure, behaviour and function.
- The QR principle of deriving symbolic models from mathematical models is of value in developing causal unit models.

CHAPTER FOUR

EXPERT SYSTEM DESIGN METHOD

4.1 HARDWARE

Potential Hardware for AI programming is discussed in Section 3.4.1. The choice of hardware for this research was largely decided by availability. Initial work was conducted in LISP on the University Harris mainframe (this LISP was actually written in FORTRAN!) and later transferred to Birmingham University's DEC20 INTERLISP. At this time the new LISP standard, COMMON LISP, had just been formulated [Steele 1984] and became available for the IBM PC/XT and AT. It was suggested that one of these machines should be acquired for this research and an IBM PC/AT was obtained, along with Gold Hill Computers Inc. Golden Common Lisp (v1.0).

In 1987 the University purchased a DEC Cluster system, supporting a version of COMMON LISP and the expert system building language OPS5. Although some development work was carried out on this machine, a mainframe is not the ideal tool for building expert systems, and it was felt that the range of alternatives provided by OPS5 was too restricted for a general exploration of the problem at hand.

The ideal tool for this kind of work is a personal workstation, unfortunately one was not available.

4.2 SOFTWARE

There are two basic choices: programming language or Expert System building tool. LISP was chosen for the following reasons:

1) It is by far the most dominant vehicle for AI work, particularly in the USA, and has been for thirty years.

2) LISP is well suited to the mechanics of knowledge representation and processing (3.4.2 & 3.5)

3) Although a PROLOG program can, in theory, do anything that a LISP program can, it is far easier to maintain complex, user defined symbol structures in LISP. This was felt to be an essential requirement as it was soon determined that the symbolic representation of process plant by computer is the key to reasoning about it. PROLOG, of course, has the advantage of an in-built inferencing mechanism, but it was preferable not to impose any one reasoning method on the research.

4) Many Expert System building tools are written in LISP or PROLOG. Each tool imposes its own way of doing things on the system builder, although alternatives may be provided. In order to retain maximum flexibility this option was rejected. At the beginning of this research the only Expert System Shells available were either too rudimentary or too expensive.

4.3 GOALS OF THE RESEARCH

This research was initiated by the suggestion of David Lihou [Lihou 1980c] that it might be possible to identify general rules for the determination of causes and consequences of deviations when conducting a HAZOP. In the context of the rapid upsurge in Expert Systems, the initial direction of the research was formulated, namely rule-based HAZOP synthesis. For reasons which are discussed in detail in Chapter 5 the research evolved into more global consideration of the generic problem of fault propagation in process plant.

4.4 EXPERT SYSTEMS DESIGN METHODOLOGY

Although the field of Expert Systems is still a very young one, for the development of complex systems two design principles emerge [Hayes-Roth 1983]:

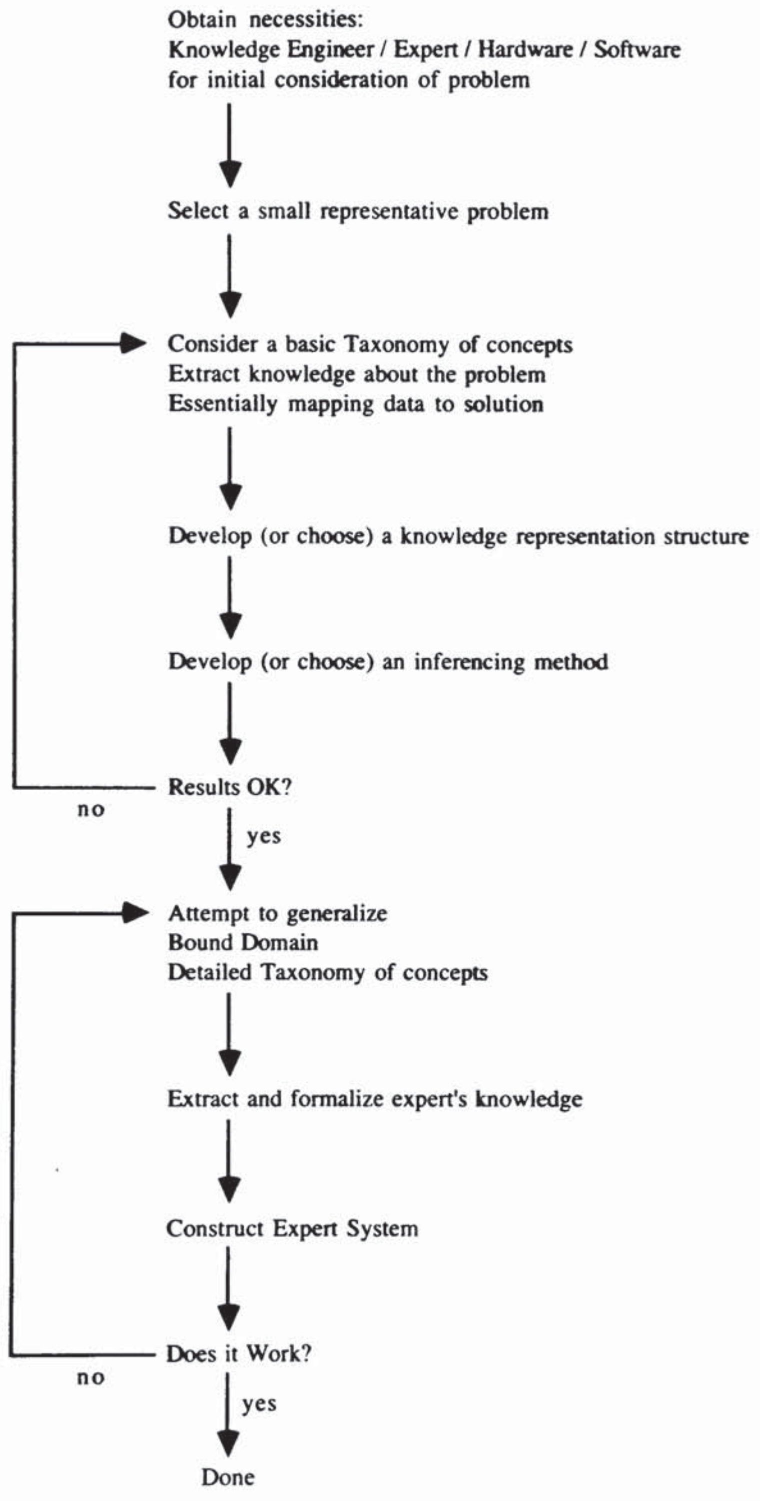
1) The process of design is an iterative one.

2) The initial goal should involve attempting to solve a small representative problem in the domain to a high degree of competence, rather than trying to capture all the domain knowledge at once. After a small problem has been tackled satisfactorily, generalization may proceed. (A disadvantage of this approach is that a smaller problem may have proportionately more unknown boundaries).

These principles are illustrated in Figure 4.4a

For the above reasons a small, representative problem was selected. This was the Ammonia Let-Down HAZOP [Lihou 1982d] which is reproduced in full in Appendix 1. The process of designing an Expert System to solve this problem certainly proved to be an iterative one. In outline, this was a gradual division of the problem into more manageable sub-problems. The evolution of the proposed solution is discussed in Chapter 5.

FIGURE 4.4a A Flowsheet for Expert System Design


Obtain necessities:
Knowledge Engineer / Expert / Hardware / Software
for initial consideration of problem
Select a small representative problem
Consider a basic Taxonomy of concepts
Extract knowledge about the problem
Essentially mapping data to solution
Develop (or choose) a knowledge representation structure
Develop (or choose) an inferencing method
Results OK?
no
yes
Attempt to generalize
Bound Domain
Detailed Taxonomy of concepts
Extract and formalize expert's knowledge
Construct Expert System
Does it Work?
no
yes
Done

CHAPTER FIVE

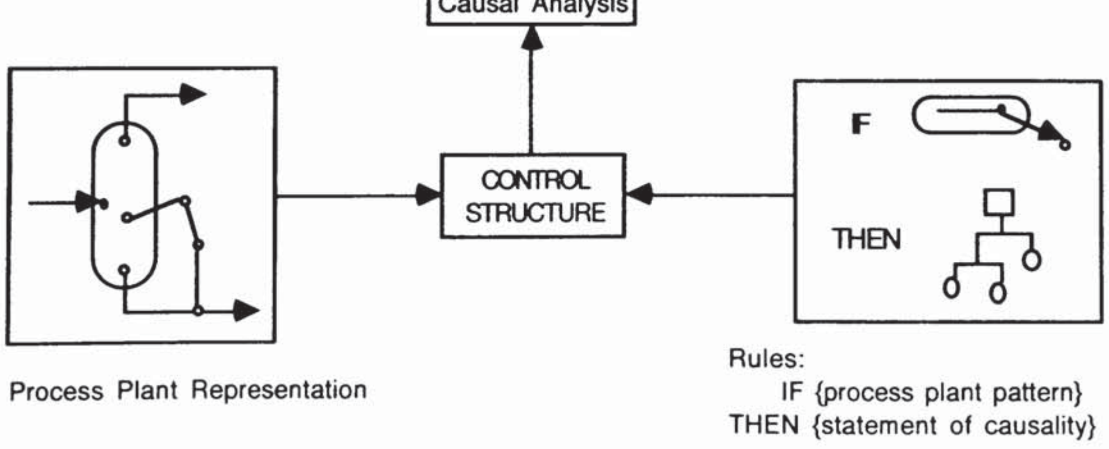
IKBS DESIGN EVOLUTION

5.1 INTRODUCTION

As discussed in Chapter 4 it is generally accepted that, in building Expert Systems of any complexity, one should first select a representative example problem and attempt to solve this problem before generalizing. With this in mind, the Ammonia Let-Down System (ALDS) was selected as an example; particular emphasis being placed on the causal analysis of the Two Phase Splitter. This item of equipment was considered to be of sufficiently low complexity to facilitate experimentation using the modest means available. The fact that the IKBS would have to be generalised meant that the entire ALDS, along with other example designs, eg Hydrocarbon Reactor and Solvay Process were also considered when building the Knowledge Bases.

The process of evolution which led to the proposed design for an IKBS which is presented in the next chapter may be considered to be a process of successive refinement of the initial conception of the solution to the problem, shown in Figure 5.1a. This involved breaking down the problem as a whole into a number of smaller sub-problems.

FIGURE 5.1.a Initial Conception of the Solution to the Problem


Causal Analysis
CONTROL STRUCTURE
F
THEN
Process Plant Representation
Rules:
IF {process plant pattern}
THEN {statement of causality}

Any rule based system for causal analysis must have the overall characteristics shown in Figure 5.1a:

1) A symbolic representation in the computer of the process plant under analysis.

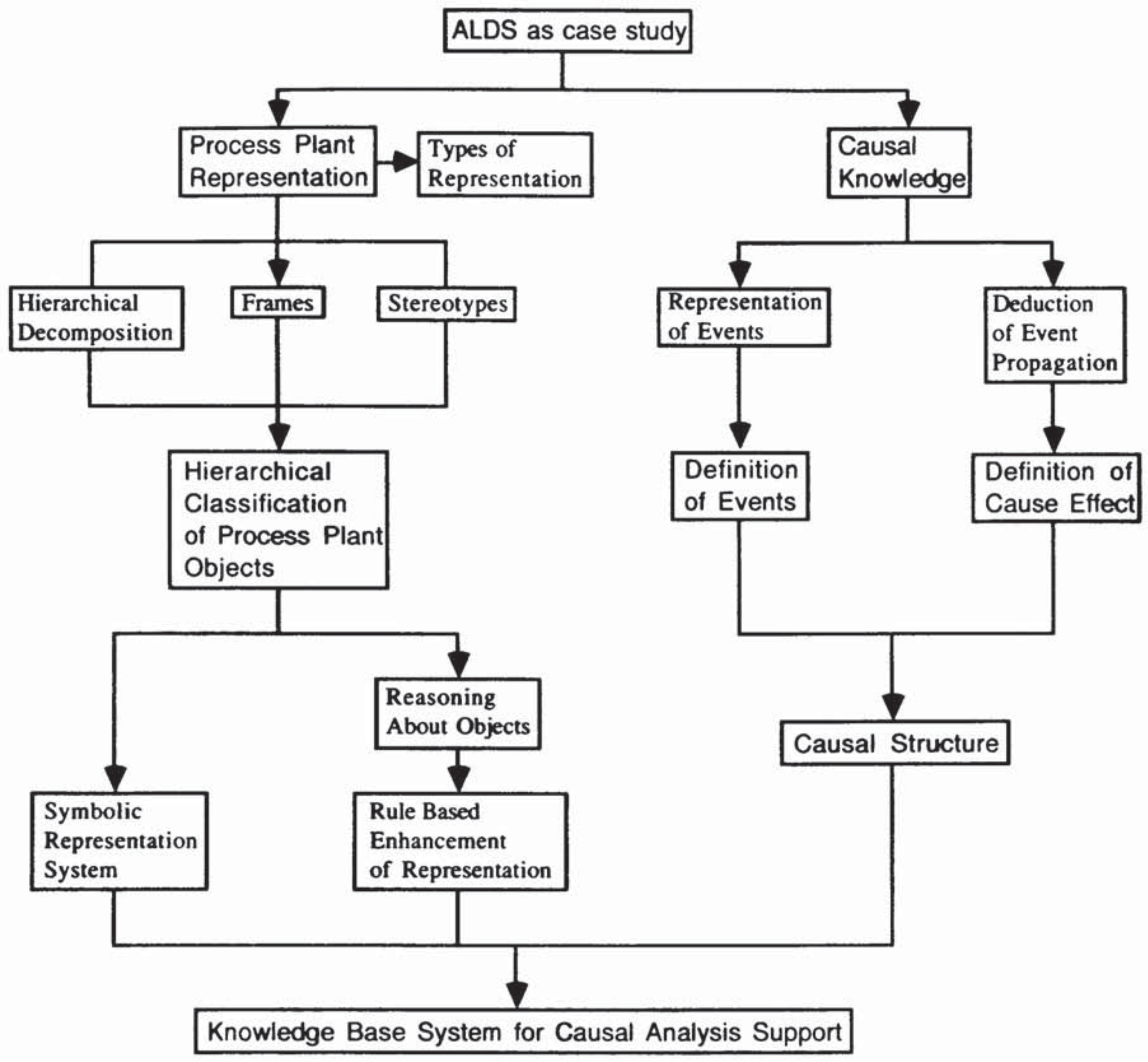
2) Rules with • conditional • A generalised pattern of a part of process plant to be matched against the plant under analysis.
   • action • Statements of events and causal links to be applied to the plant under analysis.

Figure 5.1b gives an Overview of the Design Evolution process which involves the parallel consideration of the two characteristics of process plant representation and causal knowledge as above.

Process Plant Representation

From very early on in the life of the research it was realised that the development of a systematic method of symbolic representation of chemical plant is the key to any kind of reasoning about it. This is intuitively obvious as such a problem representation is not only a requirement for describing any particular plant but, in this case, is also of considerable complexity compared to that required in many expert systems.

FIGURE 5.1b Overview of the Design Evolution


ALDS as case study
Process Plant Representation
Types of Representation
Causal Knowledge
Hierarchical Decomposition
Frames
Stereotypes
Representation of Events
Deduction of Event Propagation
Hierarchical Classification of Process Plant Objects
Definition of Events
Definition of Cause Effect
Reasoning About Objects
Causal Structure
Symbolic Representation System
Rule Based Enhancement of Representation
Knowledge Base System for Causal Analysis Support

Consider a system for medical diagnosis. Here the case-specific data for a problem to be solved may consist of a fairly simple set of statements about the symptoms exhibited by the patient; the underlying structure and behaviour of the system, ie the body functions, are taken for granted by the medic. In process plant reasoning, the case specific data must represent the structure of the plant - its P&ID, and the operating specifications: Flow rates, Temperatures, Equilibria, control system behaviour, phases and chemical species. It is also necessary to explicitly represent the functioning of the plant. Clearly this is an order of magnitude more complex than the simple medical example.

Section 5.3 discusses the development of a system for representing process plant which followed the path illustrated in Fig 5.1b. Various types of representation system were investigated and the relative merits of "High Road" or abstract knowledge and "Low Road" or detailed knowledge were considered. A means of hierarchical decomposition of the P&ID into systems and primitive objects was developed along with a frame based representation system for those objects. A system of stereotyping plant objects was investigated to represent consistently the same object in different contexts.

These ideas were combined to form a hierarchical classification of generic process plant object stereotypes, providing a systematic and manageable symbolic representation system for chemical plant. Further work involved rule based reasoning and enhancement of the representation.

Causal Knowledge

Section 5.4 discusses in detail the development of a method of causal analysis. The first requirement for causal analysis is a means of representing events and cause-effect relationships between them. A similar frame based system to that used for plant objects was developed using a classification of different kinds of events.

The central decision made concerning the derivation of causal information from a description of the plant was to split this derivation into two parts; definition of events that could occur in a given plant, and definition of the causal links between those events. Although these two tasks may involve the repetition of knowledge and are certainly more complicated than a single task, this division is justified in Section 5.4

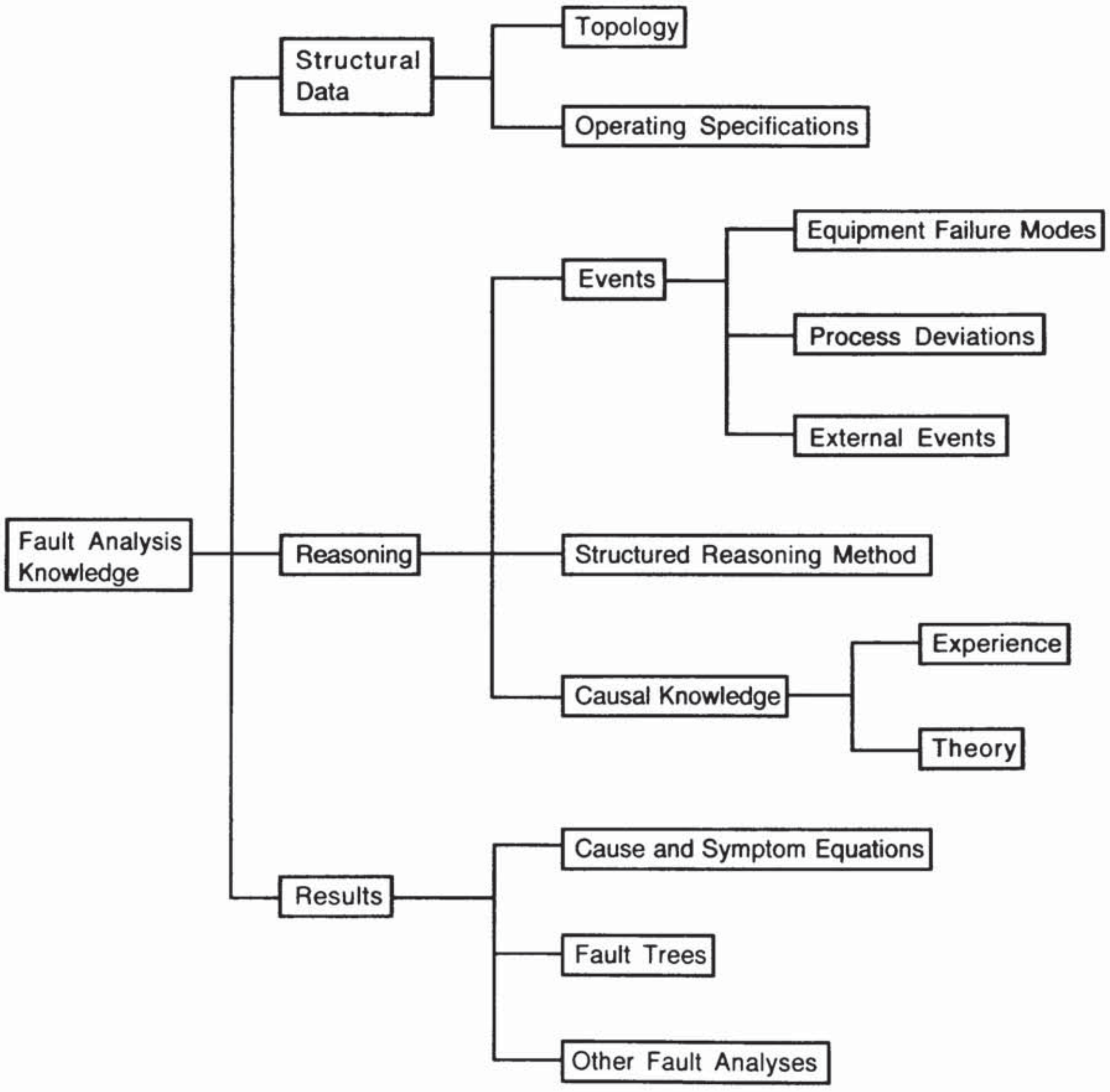
Another key decision that was made was to separate the inference of causality from structure from the process of constructing a final fault analysis. Therefore, instead of directly producing a fault analysis, a causal structure specific to the plant under investigation is produced; from this causal structure a fault analysis may be produced.

5.2 FAULT ANALYSIS

5.2.1 Fault Analysis Knowledge

The knowledge which characterises a fault analysis divides neatly into three groups: the data provided for a given analysis, the reasoning used to conduct the analysis and the end result of the analysis. These may be sub-divided further as shown in Figure 5.2.1a.

Figure 5.2.1a Fault Analysis Knowledge



Data

The first step in a HAZOP is the assembly of information relevant to the plant or design under consideration. This information, case specific data which describes the **structure** of the process, may be divided into two categories:

1) Topology

Commonly represented as a P&ID , the topology of the system is an explicit representation of the identity of plant items and their interconnections. Often implicit within this representation are assumptions relating to the behaviour of these items, for example the configuration of control devices, a control valve may be air-to-open or air-to-close. The control strategy may be implicit, eg a flow control loop may be assumed to be time independent and not pulsed.

2) Operating specification

These specifications detail the process under consideration, including:

- Materials - composition, flowrate, temperature, pressure etc
- Equipment - operating and design ratings, pressures, temperatures, materials of construction etc.
- Flow - mass transfer, heat transfer
- Control - specification and bands in the control systems

Reasoning

The reasoning used in a fault analysis consists of a structured procedure, knowledge of possible process deviations and of how these deviations are causally interdependent.

Procedure

For HAZOP a procedure is followed for considering all process items and lines and all deviations which may occur . This is detailed in Section 2.4.

Events

Process Deviations are defined by using the Property Word / Guide Word / Component system described in 2.4 (Table 2.5.4b). The particular key words used are, of course, dependent on the context of the deviation they represent, ie the item of equipment where they occur. Failure modes of items of equipment may also be drawn from tables as discussed in 2.4 and given in Appendix 2. Credibility of events is also a consideration.

Causal Knowledge

Once an event is identified it is necessary to determine by causal reasoning its causes and consequences. This causal reasoning takes place at two levels:

- Experience: A given pattern may be recognised by the expert and, drawing from experience, he is able to describe directly the causality.
- Theory: In more complex, or novel situations, recourse may be made to fundamental generic knowledge of chemical engineering operations, processes and equipment behaviour.

Sources of Fault Analysis Knowledge

Various potential sources of knowledge for constructing an expert system for fault analysis were identified:

(1) An Expert. The normal route for the construction of an expert system involves the elicitation of the knowledge of an expert. Unfortunately one was not available.

(2) An example Fault Analysis. The example selected here is the Ammonia Let-Down System (ALDS).

(3) Equipment failure mode and event tables, and decision tables, used to define the valid events that may occur.

(4) Flowsheets from the literature for conducting fault analyses, used in the formulation of the methods for producing cause and symptom equations and fault trees from the intermediate step of the causal analysis.

(5) Rules for the construction of fault trees, eg those given in section 2.5.3.

(6) Learning. The problem was considered to be too complex to even consider a learning system. In developing a learning system it is first necessary to identify a paradigm for reasoning; this is the subject of this research.

(7) Various knowledge sources were consulted and borne in mind when developing knowledge base rules, including:

(i) Case studies.

(ii) Rules gleaned from the literature.

(iii) Fault Tree Synthesis Methods - manual (2.5.3)

- automated (2.5.4)

(iv) Fault Diagnosis Methods - (2.7)

(v) Background Knowledge of Chemical Engineering Operations.

5.2.2 **Manual Fault Analysis**

Fault Analyses must meet two criteria:

1) True: The analysis must be a faithful reflection of the way in which the real world chemical plant behaves.

2) Veridical: The analysis must be <u>internally</u> consistent, following a logical and consistent method of describing causality. This is critical if the analysis procedure is to be copied by an automated system.

In practice, any number of experts who tackle the same problem will all produce differing analyses.

The veridicality of the example Cause and Symptom Equation (CSE) HAZOP - the Ammonia Let-Down System (ALDS), has been subjected to particular scrutiny in an attempt to elucidate the structure of the reasoning behind the analysis. This close examination reveals that, in common with other analyses, the expert very much "makes it up as he goes along". Each equation that is developed relies on the context of the previous equations. The <u>meaning</u> of each event described in the equations is also strongly dependent on context. These points are illustrated in the analysis of the Cause Equations for C1, the Two Phase Splitter in the ALDS, the paper containing this is included as Appendix 1:

C1 Cause Equations

1

$$\text{C1 (level no)} = \overset{\boxed{1}}{\text{LT1(stuck high)}} + \overset{\boxed{2}}{\text{LCL1(giving more flow)}} * \overset{\boxed{3}}{\text{LAL1(FD)}}$$

$$\overset{\boxed{2}}{\text{LCL1(giving more flow)}} = \overset{\boxed{4}}{\text{LIC1(set low)}} + \overset{\boxed{5}}{\text{LCV1(open too much)}} + \overset{\boxed{6}}{\text{V3(open)}}$$

$$\text{LAL1(FD)} = \text{LSL1(set low or stuck high)} + \text{LAL1(failed)} + \overset{\boxed{8}}{\text{LAL1(ignored)}}$$

2

$$\text{C1(level less)} = \text{LT1(indicating high)} + \overset{\boxed{4}}{\text{LIC1(set low)}} + \overset{\boxed{6}}{\text{V3(open)}}$$

3

$$\overset{\boxed{7}}{\text{C1(level more)}} = \overset{\boxed{7}}{\text{L2(blocked)}}\text{+LT1(ind low)+(LIC1(set high)+}\overset{\boxed{7}}{\text{L2(BV)}}\text{+L2(RV))*}\overset{\boxed{3}}{\text{LAH1(FD)}}$$

$$\text{LAH1(FD)} = \text{LSH1(set high or stuck low)} + \text{LAH1(failed)} + \text{LAH1(ignored)}$$

$$\text{L2(BV)} = \text{V1(blocked)} + \text{LCV1(blocked)} + \text{V2(blocked)}$$

$$\text{L2(RV)} = \text{LCV1(insufficiently open)}$$

(1) Stuck High is not in the Failure Mode Table.

(2) The condition LCL1(giving more flow) is not clearly defined. In this context it actually means "giving too much flow", compared to the desired equilibrium value with respect to the inflow to C1. LCL1(giving more flow) would, in fact, be caused by LT1(stuck high).

(3) LAHL1 as defined on the P&ID has been split into two devices for the purpose of the analysis: LAL1 and LAH1.

(4) LIC1 (set low) would most likely cause (C1 level less). LIC1(set zero) or LIC1(set low low) would perhaps be more appropriate here. The fault analysis does not distinguish between those conditions such as LIC1(set low) which would give a new but stable condition, and LT1(stuck high) and V3(open) which would lead to an unstable plant condition until C1 is empty.

(5) In this context, LCV1(open too much) is used as a primary failure. Taken in isolation, the event LCV1(open too much) may be thought of as a command fault,

perhaps caused by a control loop failure. The correct failure mode here is LCV1(stuck open more), which is a primary failure of the valve and not a command fault.

(6) For cause equation 1, V3 is treated as a part of the control loop LCL1, which is not really the case. Additionally, V3 may fail open to varying degrees of internal leakage. The control loop would then either:

 (i) Be capable of compensating for the leakage by throttling back LCV1, in which case the level in C1 will be maintained at the set point.

 (ii) Be incapable of compensating sufficiently, ie LCV1 will close and the vessel will empty.

 V3(open) is therefore unlikely to cause C1(level less) as in cause equation 2, and would rather result in no change to the level or zero level.

(7) C1 level may change in the following ways:

 (i) Decrease to, and then stay at, zero.

 (ii) Stabilise at a lower level.

 (iii) Stabilise at a higher level.

 (iv) Increase to a maximum, where liquid enters the gas offtake.

 The guide word system in the ALDS paper does not allow for condition (iv), but only for C1(level more). (iv) might be termed C1(level max).

 L2(BV) and L2(B) will cause C1(level max) whereas LT1(indicating too low) will cause C1(level more).

 The failure LT1(stuck low) is not included here and would cause C1(level max).

(8) The human factor is introduced here as it is assumed that if LAH1 is not ignored, an operator will intervene to prevent the condition C1(level no).

5.3 THE REPRESENTATION OF PROCESS PLANT

5.3.1 **Introduction**

This section is concerned with the development of a systematic method of representing the **structure** of a chemical plant. That is, a plant is considered to consist of a number of interrelated physical **objects** - vessels, lines, valves, control devices, materials etc. The development may be looked at in four ways:

1. Alternative Representations of process plant objects.
2. Establishment of stereotypes - generic objects to be instantiated according to each particular occurrence of that type of object.
3. Levels of abstraction - consideration of high level or low level knowledge and the possibilities of symbolic modelling - discussed in section 5.5.
4. Reasoning about plant structure using rules - discussed in Chapter 6.

This section (5.3) is concerned with the evaluation of the symbolic representation system as in (1) and (2) above.

5.3.2 **Alternative Representations**

It is clear that a chemical plant is far too complicated to be adequately described by a loose set of assertions. The topology of the plant suggests that these assertions should be grouped to reflect the different elements in the plant and must describe their physical interactions. For this reason, a Frame Based system was chosen (Frames are discussed in section 3.5.2).

Attempts were made to devise frames to represent the objects in the ALDS P&ID; the Two Phase Splitter P&ID is shown in Figure 5.3.2a. Frames were devised for high level objects such as the vessel C1, lines and control systems; an example of an early frame for C1 is shown in Figure 5.3.2b. It became clear that these types of system frames are too general to describe adequately the different possible combinations of the elements that could make up the system and the range of potential ways in which they may interact; it is necessary to represent the individual devices that make up a control system for example.

FIGURE 5.3.2a Two Phase Splitter P&ID

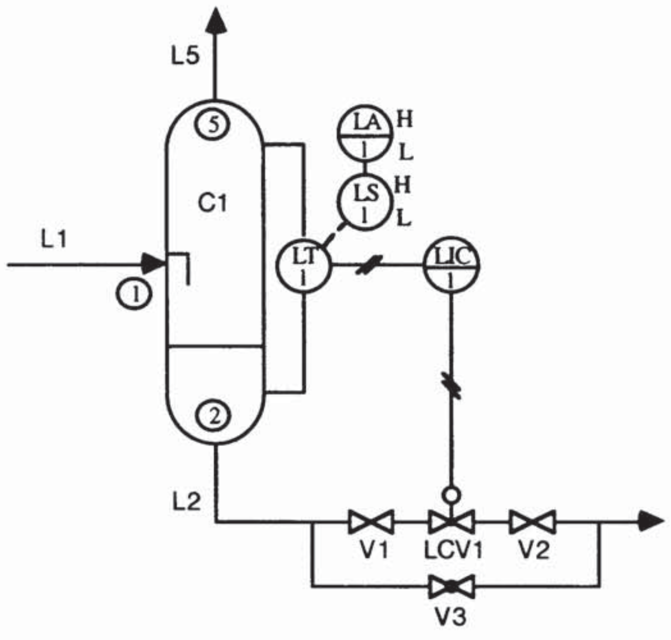


FIGURE 5.3.2b Example of Early Frame

```
(C1  (identity     (:= vessel system))
     (boundary     (inflow     (:= L1))
                   (outflow    (:= L2 L5)))
     (contains     (bulk phase       (:= G1 L1))
                   (bulk interface   (:= G1 L1))))
```

(here G1 and L1 refer to process fluid bodies of gas and liquid respectively)

The decision was made to define vessels, lines and control systems as **systems**, which have a frame representation at this level, but which also contain objects detailing the internal structure of the system. For example a line system contains valves which are joined together by flow paths.

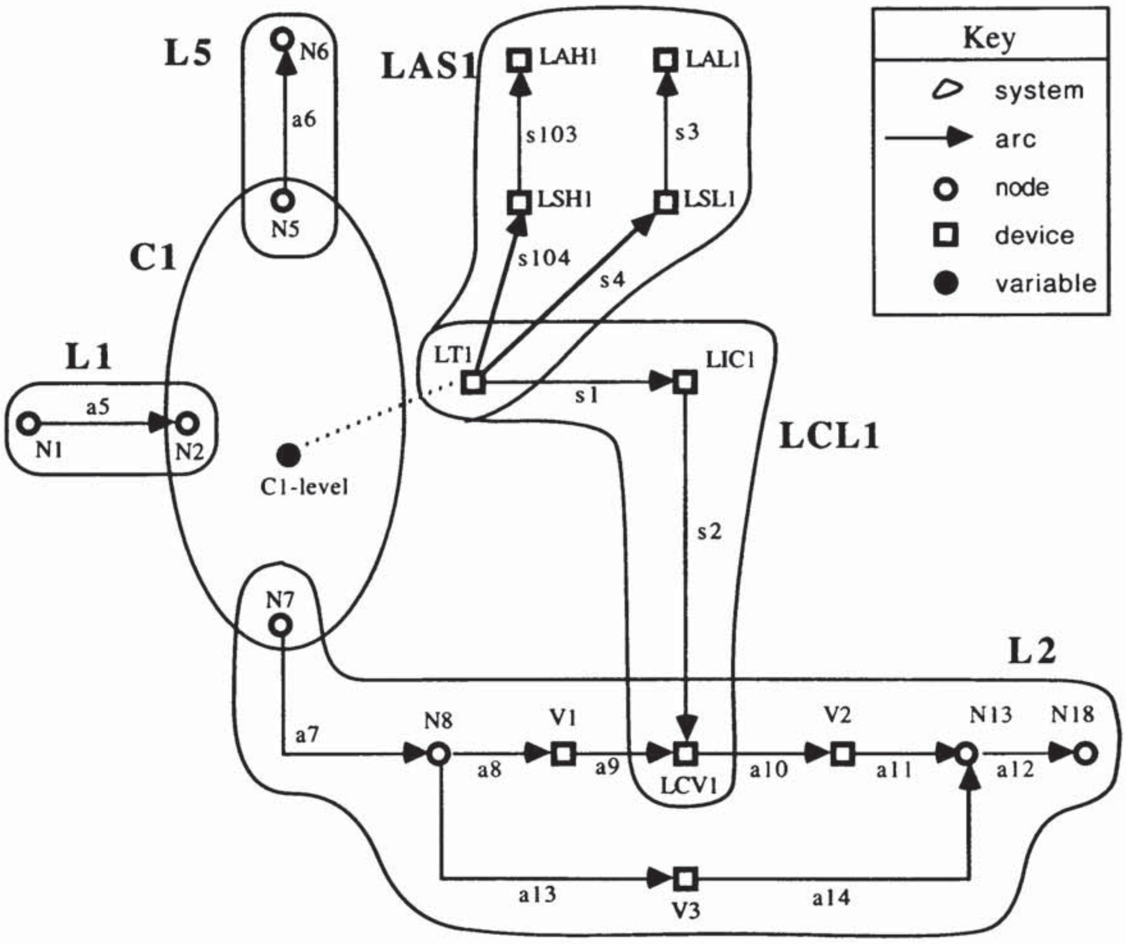
This suggests the possibility of a hierarchical decomposition of a P&ID into systems, sub-systems and the objects contained within them, each of these to be represented by a frame. There are a number of problems to be solved, principally;

1) How to represent the internal processes of vessels ? This is clearly more complex than representing the individual objects that comprise a control system or a line.

2) How to represent the interconnections between objects ?

Consideration of these questions led to the establishment of three principal **object types**: system, node and arc. Systems are as above, containing nodes at places of interest, these nodes being interconnected by arcs. These three basic types of object are reflected in the final hierarchical decomposition of the Two Phase Splitter shown in Figure 5.3.2c.

FIGURE 5.3.2c Hierarchical Representation of the Two Phase Splitter



a = material flow arc

s = signal arc

5.3.3 **Stereotypes**

Once a particular set of frames was worked out for describing objects in the ALDS it became necessary to generalise these representations. This involves the construction of **stereotypes**, generic objects stored in a permanent library to be instantiated to represent objects in a specific case.

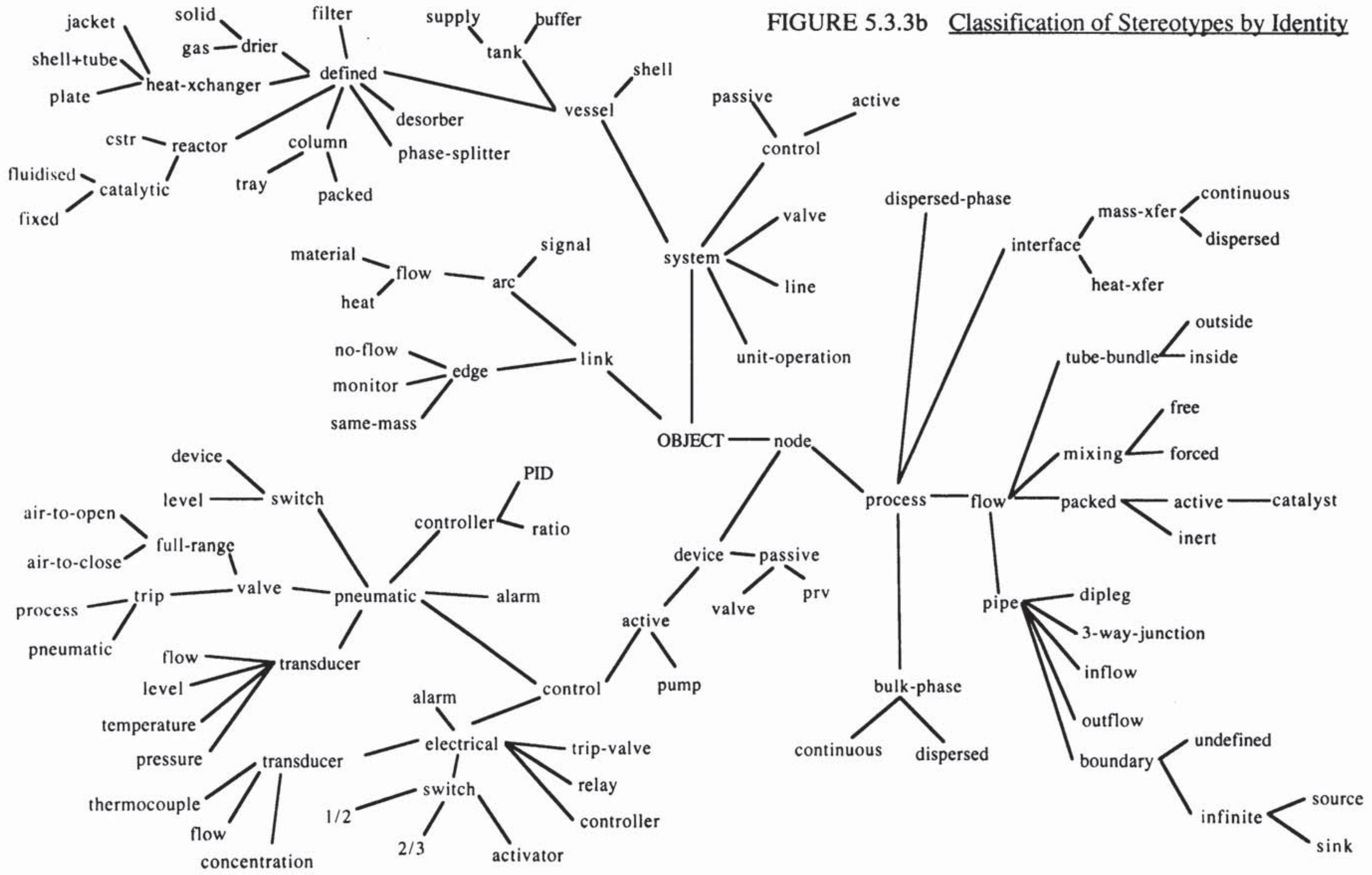
The construction of generic or stereotype frames for the specific case of the ALDS does not, in itself, present any serious problems. The question arises, though, of how to organise and manage the different object stereotypes. This organization centres on the identification of each stereotype as being a unique generic object. The initial scheme was to have a library of completely independent stereotype definitions. Each definition was uniquely identified or mapped by two slots - the TYPE slot and the IDENTITY slot; examples of this kind of stereotype are given in Figure 5.3.3a.

FIGURE 5.3.3a Type / Identity Stereotypes

```
(C1   (type      (:= system))
      (identity    (:= vessel)))

(LCV1 (type      (:= device))
      (identity    (:= control valve)))
```

For the management of any library of stereotypes it is desirable to classify these stereotypes. An example of an early classification using the type and identity slots is given in Figure 5.3.3b.

FIGURE 5.3.3b Classification of Stereotypes by Identity



It is clear from these attempts to classify the different types of process plant objects that a number of natural classifications may be incorporated. For example, a control device may be a sensor, a processor or an actuator. A sensor, in turn, may be a switch, a level transmitter, pressure transmitter etc. From experimentation with different classification methods it was decided to drop the identity slot system in favour of a full hierarchical classification of process plant objects based on their **type** designation. The final hierarchical classification which was decided upon is shown in Figure 5.3.3c.

For clarity, only those objects within the ALDS system are desribed in the hierarchy (eg no temperature variable is given). In addition:

(1) Inheritance may take place through more than one path, eg a valve is both a line-node and a device.

(2) An alarm is included as an actuator as it may elicit manual intervention.

(3) LINE-ARC refers to arcs which represent potential material flows.

(4) COMPOUND refers to a chemical component.

The representation of the hierarchical classification of stereotypes and the attribution of properties to these stereotypes is discussed in Chapter Six but it is relevant to present some of the benefits of this approach.

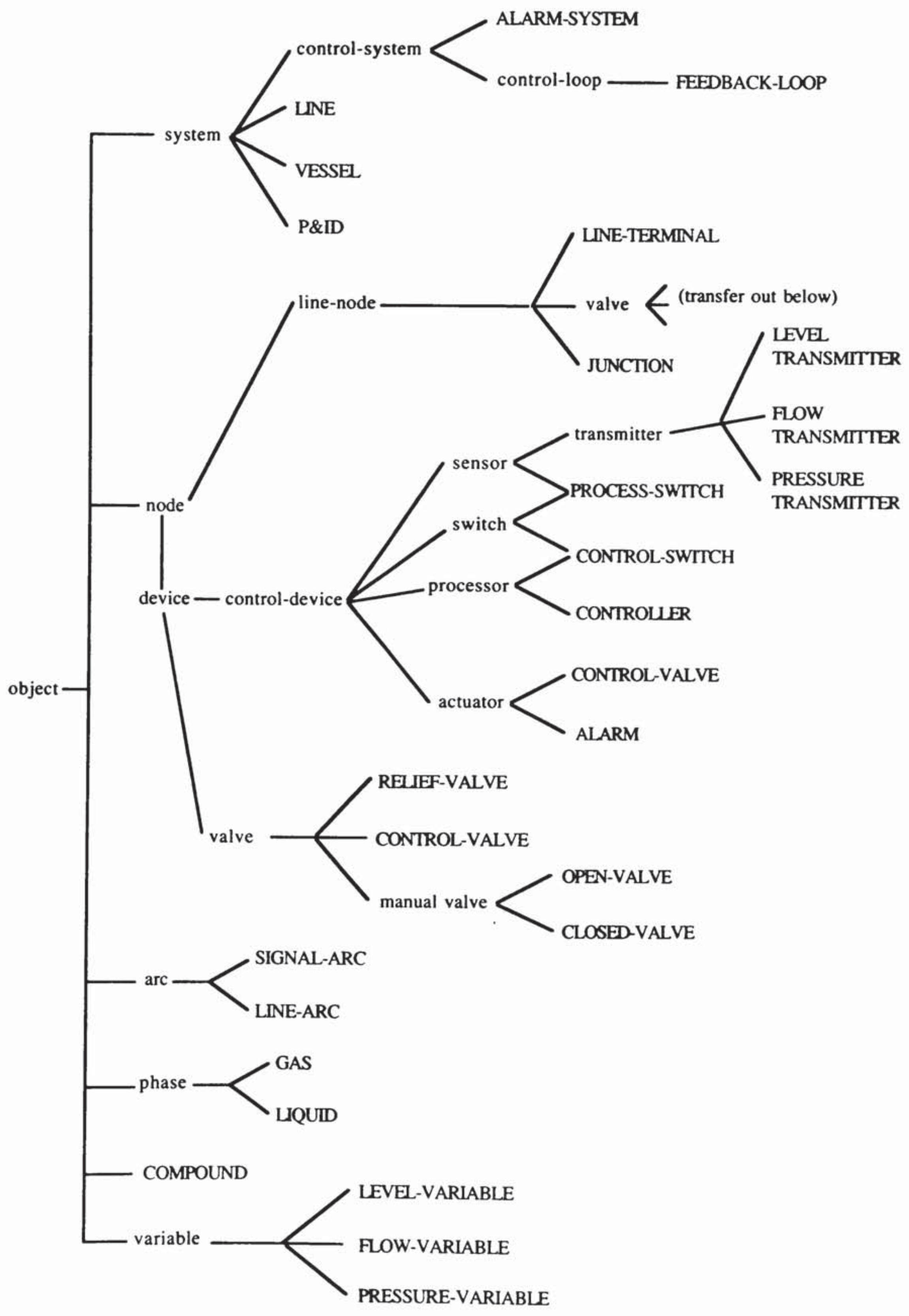
1. <u>Completeness</u>

   Any specific object to be represented is described in the hierarchy, in relation to similar objects of its type. This helps in achieving a representation system capable of describing all plant objects and in accommodating new stereotypes in an organised fashion when the representation is generalised to include other plant.

2. <u>Inheritance</u>

   In frame based systems terminology, the hierarchical classification is the first step in an IS-A HIERARCHY. We may define, for example, a stereotype frame for a sensor. All objects which are subclassified as of type sensor, such as transmitters and switches, may inherit attributes from the sensor frame. This is invaluable in constructing an organised system for attribution of objects, and in ensuring the completeness of the representation of any particular plant.

FIGURE 5.3.3c Hierarchical Classification of Process Plant Objects



5.4 CAUSAL REPRESENTATION AND KNOWLEDGE

5.4.1 The Representation of Causality

From a consideration of the events that may occur in a chemical plant the following event type classification has been developed:

Type EVENT is of sub-type:
- FAULT
- FAILURE
- DEVIATION
- GLITCH

There are three kinds of causal relationships between events:

Causation

eg e1 -> e2 Direct expression of cause-effect

Constraint

eg e1 <-->e2 Events e1 and e2 are bound to occur together

Event Set

eg E = {e1, e2, e3}

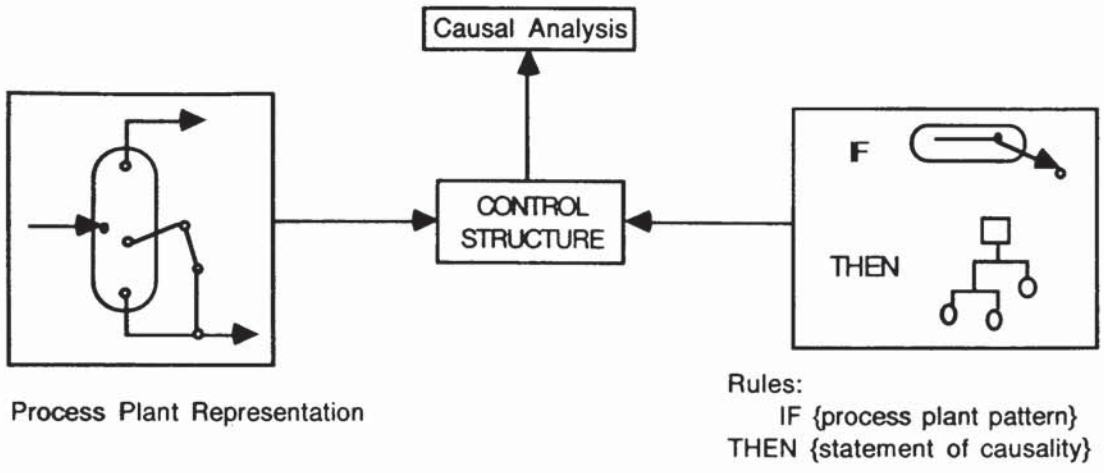
The event E is said to represent a set of process conditions which covers the more detailed events e1, e2, e3. All these events are constrained.

5.4.2 Causal Knowledge

5.4.2.1 Determining Cause-Effect from Structure

In order to produce statements of causality from a structured description of the process it is necessary to define rules which map the relationship of the **object space** to the **event space**. The overall picture of this is shown in Figure 5.4.2a.

FIGURE 5.4.2a Rules Deriving Causality from Structure


Causal Analysis
CONTROL STRUCTURE
F
THEN
Process Plant Representation
Rules:
IF {process plant pattern}
THEN {statement of causality}

The **Search Space** in a rule based system is mapped out by considering all possible ways in which the rules can link up in going from data to goals. This linking is defined by the transitivity of the rules:

IF A THEN B

IF B THEN C -> Given A, infer C by transitivity

The entire search space for rules of this kind may be visualised as a network or tree, cf decision trees which are sometimes created in the construction of an Expert System; the tree is subsequently transformed into rules. Construction of the decision tree or mapping out the entire search space is the only way to deterministically ensure the desired answer for all problems.

The rules for generating causality from structure, as in Figure 5.4.2a are of the form:

IF A THEN x ---> y

IF B THEN w ---> y

Here A. B are structure patterns, x-->y is a statement of causation, x, y being events. The search space for these rules is determined by the event space, or causal structure created by the consequents of the rule. Ordinary rule transitivity may occur if the antecedent of the rule is an event space pattern, eg {IF x->y then y->z}; this will produce more causal relations. If the other two types of causal relation described in 5.4.1, constraint and event set, are also included the event space becomes very complex indeed. Early experimental KBSs using the Two Phase Splitter example produced of the order of seven hundred events.

The conclusion that may be drawn from this discussion is that it is impractical to attempt to deterministically evaluate the action of the rules, ie to attempt to produce a pre-ordained solution (here the ALDS CSEs) from a structural description. Even if this were done, there is no guarantee that the rules so formed would be readily generalised - it is obviously beyond practicality to map out all possible search spaces for all possible problem PIDs. The rules so formed are likely to be distorted by the requirement to reach the desired exact solution.

The implication of these conclusions is that it is extremely difficult to produce directly a fault analysis using a rule based system by simply constructing a rule base. Generally, expert systems work by inference and not determination as the search space is too large to be mapped out. The approach to this problem involves constructing an IKBS that is thought to be capable of expressing the knowledge needed to solve the problem and then successively refining that knowledge. The implications of this discussion on the goals of the research is discussed in Section 5.5.3.

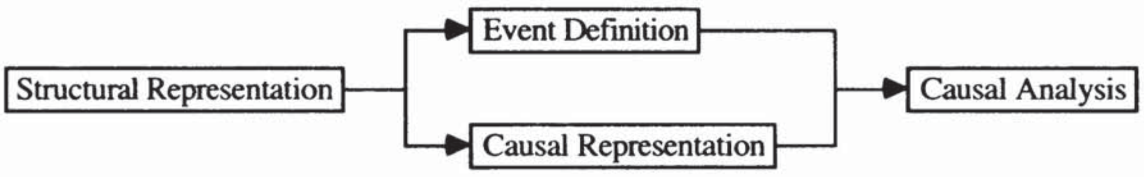
The solution adopted to the search space problem is to construct an event space or **causal representation** and to derive the causal analysis from this. This representation consists of events defined by the rules, interrelated by the three causal links - Causation, Constraint and Event-Set. The production of a given fault analysis from this representation is discussed in the next section (5.4.3).

If the causal representation is not deterministically produced then it is very difficult to ensure its correctness. One way of doing this is by refining the rule base but this process can only go so far. Another approach is to try to structure the causal representation according to the particular chemical plant under consideration. One way of doing this is by the prior definition of the events that may occur in a plant. This is explained in the following section.

5.4.2.2 <u>Event Definition</u>

In a HAZOP, events that may occur in a design are defined, using keywords, prior to any consideration of cause-effect. The existence of these keywords along with some experimentation suggests that it is considerably easier to identify those events which may occur in a plant than to determine how these events will propagate. This is the approach adopted in the proposed design described in Chapter Six; the possible events that may occur are determined separately from possible causal relations, ie the production of the fault analysis, proceeds as in Figure 5.4.2b.

FIGURE 5.4.2b <u>Overview of Causal Analysis Production</u>



This prior determination of all events that may occur in a process is linked to those events which will appear in the final fault analysis. This has two principal advantages:

1. All events that must appear in a comprehensive fault analysis, such as HAZOP using cause and symptom equations, are known.
2. Duplication of knowledge actually occurs because it is necessary to mention events again in the causal rules. However, as valid events are known, the causal rules may be generalised without having to ensure that the events they generate would actually occur, eg it is possible to relate all effects of a valve position without regard for the fact that a closed valve cannot fail closed; the event failed closed will not appear in the final fault analysis as it is not in the Valid Event Set.

5.4.3 **Production of the Fault Analysis**

The production of the Fault Analysis in the proposed design is described in Section 6.6. The analysis is produced by assisting the analyst using the Structured Representation, Valid Event Set and Causal Representation. However, other possibilities have been examined. One of these is to use general rules to assist in the elicitation of applicable cause-effect from the Causal Representation, by a process of Heuristic Search.

Heuristic Search

The use of heuristic search in rule based systems is discussed in Chapter Three.

Experience suggests that, in deciding upon the causes of a given event, an expert analyst makes an "educated guess" at those causes. Should a given cause prove not to be straightforward, the analyst may resort to detailed forward tracing of cause-effect to prove the link.

The "educated guesswork" as to possible causes is the kind of expertise expressed in the rules for CSE HAZOP developed by Lihou, illustrated in Figure 2.4.5b. These rules essentially describe where to look for the causes of a particular event. This suggests that they may be employed for heuristic search of the causal representation. The procedure would be something like this:

- Select an event for which causes are to de determined.
- Employ heuristic rules to identify the location and type of the possible causes.
- Use the Valid Event Set to narrow this list of possible causes.
- Use the Causal Representation to establish propagation from the candidate causes to the selected event.

Gradual degradation of the performance of the system could be included by reliance on any other fault propagation revealed by the Causal Representation, ie by considering fault propagation from events which have a causal link, but which are not identified by heuristic rules.

5.5 DISCUSSION

5.5.1 Levels of Abstraction

The different kinds of knowledge used for solving a problem in a particular domain may be considered to occupy an abstraction spectrum. At one end of the spectrum is the highly detailed theoretical knowledge that underpins the domain, and at the other end of the spectrum is knowledge at a high level of abstraction, heuristic knowledge which makes large-scale generalizations about the domain. These two types of knowledge may be referred to as "low road" or detailed knowledge, and "high road" or abstract knowledge, respectively. The type of knowledge which characterises the expert approach to the problem is generally the more abstract or high road knowledge.

The methods developed for automated FTS described in Section 2.5 are all characterised by their use of detailed or low road knowledge. Exclusive use of low road knowledge has here been demonstrated to be unworkable as, to put it simply, "the whole is more than the sum of the parts". For example, it is a very laborious process to deduce the dynamic behaviour of a control system, particularly under fault conditions, from a consideration of the behaviour of the devices and the process flows that make up the control system. Lapp and Powers, for example, get around this problem by using abstract rules governing the global behaviour of control systems to guide their FTS algorithms.

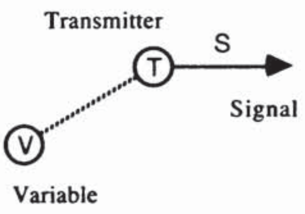
In the context of the problem at hand, abstract knowledge may be considered to be that referring to the overall behaviour of systems and items of equipment: The expert looks at a piece of equipment and, from experience, is able to write down the general behaviour of the equipment. However, in a novel situation or a particularly difficult one the expert may fall back on the underlying theory behind chemical engineering processes and use a study of the detailed behaviour in order to arrive at a picture of the general behaviour.

It follows from this discussion that neither abstract nor detailed knowledge is, on its own, sufficient to solve the problem. It is desirable to use abstract knowledge as it is, by nature, more economical, but in situations where this knowledge is not applicable, detailed knowledge of cause-effect is necessary. This duality of knowledge is also necessary in meeting an important characteristic of successful Expert Systems - they degrade gradually at the boundaries of their capability rather than just say "don't know".

5.5.2 **Symbolic Modelling**

If all the items that comprise a chemical plant are considered as objects, then to deduce the behaviour of the plant it is necessary to have behaviour models of the objects themselves. This is the methodology which underlies the component model based methods of Fault Tree Synthesis described in Section 2.5.3. As an example, a model of the behaviour of a transmitter might be as shown in Figure 5.5.2a. This model describes the input / output behaviour of the device (quantity space {-, 0, +}) which is conditional upon the normal or failed state of the device. The model has the advantage that it also defines the modes of failure of the device.

FIGURE 5.5.2a Partial Model of a Transmitter



T= normal

V	S
-	-
0	0
+	+

T= indicating too high

V	S
-	0
0	+
+	+

T= stuck high

V	S
-	+
0	+
+	+

ETC

This representation of a device as an object with a clearly defined set of inputs and outputs suggests that a symbolic simulation of an object may be created; given input variables and the object state, the simulation would produce the outputs. This could be implemented using Object Orientated Programming, as discussed in Section 3.5.2.

In this paradigm the behaviour of each object is defined by its "message handler". In the case of the Transmitter in Figure 5.5.2a, the message handler would regard the inputs of the device (ie the variable value) as incoming messages, and would send the appropriate output message to the device to which its output signal is connected. Messages are instructions that are sent between objects and between the user and objects. Causal propagation is achieved by using messages. Messages may be sent by an executive to set fault states in the symbolic model. If fault propagation is initiated at one point in the model, it is passed on by messages from one object to another. This effects a symbolic simulation of the chemical plant.

Although this methodology has the attraction of being a purely modular approach, it was not explored further for a number of reasons:

1. Difficulty in developing generic object models which are context free. It is also difficult to develop some objects at all, particularly for vessels where the process may involve very detailed analysis of the behaviour of the vessel.
2. A symbolic model of this nature would work from cause to effect only. A failure is chosen and this failure propagates through the model. In order to determine the causes of a particular event it would be necessary to consider all possible fault states of the model, of which there would be an unmanageably large number.
3. The methodology is too "low road" or detailed in nature and suffers from consequent problems of complexity and opaqueness (ie cannot see the wood for the trees). A more abstract level of knowledge is required.
4. All the information present in the symbolic models may be represented as rules. As some rules are necessary anyway it makes sense to use rules only.

5.5.3 **The Goals of The Research**

As mentioned previously (Chapter 4) the goal of the research has evolved from its initial proposal to automate HAZOP in the form of logical equations using expert systems methods. There are a number of reasons for this:

1) It is clear from an analysis of the literature that the problem of fault propagation in process plant is a generic one [Andow 1980b], facing similar problems and tackled by similar approaches. An analyst who is generally proficient in, for example, HAZOP, is likely to be reasonably adept at similar techniques: FTA, FMEA, ET, CCD and Fault Diagnosis. This implies that a means of automating any one of these analyses would form a suitable basis for tackling them all.

2) The process of conducting a fault analysis provides the analyst with a deep understanding of the system under consideration. The reputation, experience and proven competence of the expert analyst provides some guarantee of the completeness and accuracy of the analysis; this is very important as these techniques are usually carried out in the context of high risk operations. For these reasons, even if fully automated synthesis were possible today, full reliance would be unlikely to be placed on it for the foreseeable future. Therefore there is a need for <u>assistance</u> in fault analysis.

3) The nature of expert systems is such that they operate by inference rather than determination. In other words, they attempt to locate the <u>most likely</u> solution to a problem. In fault analysis it is necessary to have a high degree of confidence in the correctness of the analysis; any expert system would have to have a proven track record before it became acceptable. The only way to have a deterministic Expert System is to map out the entire search space (eg using a decision tree - Section 3.5.2) which, for any real problem, is too large to be practicable. Search space considerations are discussed in Section 5.4.2

4) As may be seen from the critical look at fault analyses in section 5.2.2 any manual analysis must meet two basic criteria:

   i) Correct - the analysis must accurately reflect fault propagation in the system under consideration.

   ii) Veridical - The analysis must be internally consistent. This criterion is not usually met as often an analyst makes it up as he goes along. This is a big problem for automated synthesis which requires a formal and structured method for conducting the analysis.

For these reasons the nature of the research has evolved from consideration of automated HAZOP synthesis to the provision of support for fault analysis in general. This involves the synthesis of a general causal structure from the initial plant representation. Interpretation of this causal structure is dependent on the nature of the required fault analysis. Where the interpretation is for HAZOP, the initial proposal is fulfilled.

The architecture of the Knowledge Based System intended to solve the problem has naturally gone through a similar process of evolution. It is characterised by the fact that fault analysis is a difficult problem even for humans. Attempts to reduce the complexity of the problem by subdivision, and by analysis of the expert approach to it, have identified that many different types of knowledge and reasoning are used. This is reflected in the final design which contains six knowledge bases representing the division of the problem as a whole into a number of more manageable sub-problems.

CHAPTER SIX

THE PROPOSED DESIGN

6.1 INTRODUCTION

This chapter describes the proposed design for an IKBS for assistance in Fault Analysis of chemical plant. In order to minimise the complexity of the description, the KBS is described as it applies exclusively to the Ammonia Let-Down System (ALDS) and in particular the Two Phase Splitter (TPP). In other words, only the representation system and rules that are necessary for the analysis of the ALDS are discussed. This chapter gives an overview of the design followed by a description of the various stages involved, from the initial description of the plant through to the fault analysis:

6.2 An overview of the architecture of the proposed IKBS.

6.3 Describes the representation of the structure - topology and operating specifications - of a chemical plant.

6.4 Describes the definition of the events that may occur within a process.

6.5 Describes the inference of causal relations between events in the process.

6.6 Discusses the production of the fault analysis based on the previous stages.

6.2 OVERVIEW OF PROPOSED ARCHITECTURE

Figure 6.2a gives a pictorial overview of the architecture of the proposed IKBS.

From Figure 6.2a there are 5 stages that may be identified:

(1) Structure Input

Case specific data, here for the Two Phase Splitter, is input by the **user** according to an **initial structure syntax** to produce an **initial structure** representation. The syntax consists of a set of generic, stereotype frames which are instantiated to represent items in the process.

(2) Structure Enhancement

The **initial structure** representation is enhanced and completed by **structural rules** which produce a **final structure** representation which is in accordance with the **main structure syntax**. At this stage the physical description of the plant in the computer is considered complete.

(3) Event Definition

From the **final structure** representation of the process, **event definition rules** and **generic event definitions** are used to determine a **valid event set**: those events that may occur within a process that an analyst would consider "significant"; these events appear in the final causal analysis.

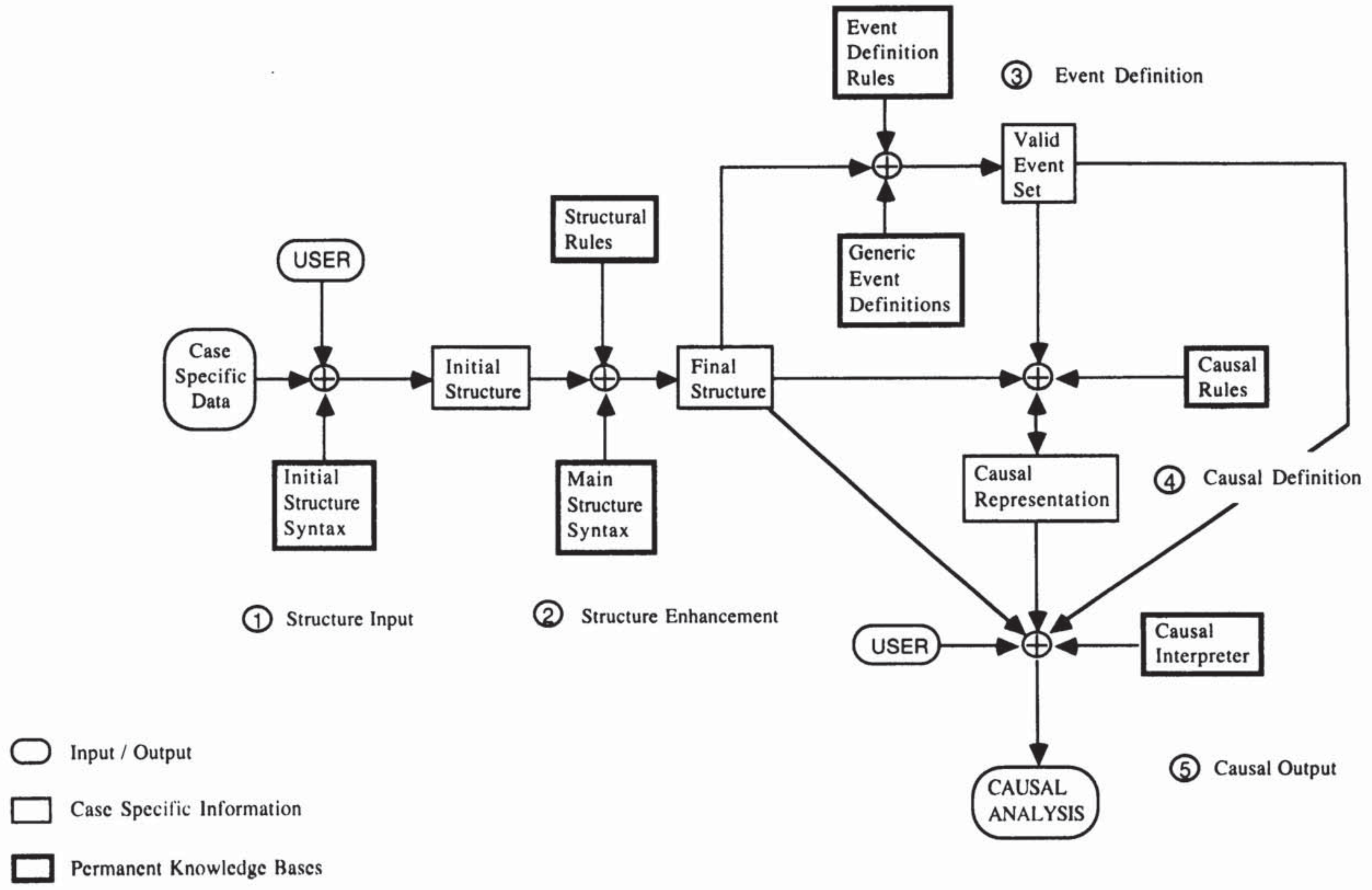
(4) Causal Definition

Causal rules are applied to the **final structure** and the **valid event set** to produce a **causal representation** which reflects the propagation of faults within the plant. This data structure consists of a network of events connected by causal relations.

(5) Causal Output

The **user** is assisted by a **causal interpreter** which uses the **final structure**, the **valid event set** and the **causal representation** in producing the **causal analysis**. The causal interpreter is one of a number of modules, each of which is designed to cope with a particular form of fault analysis. There is one for Fault Tree Analysis, one for HAZOP, one for FMEA etc.

FIGURE 6.2a The Proposed Architecture



Of great importance is the idea of a syntax for knowledge representation, to which any given piece of knowledge must conform; this is a similar idea to the establishment of the central data structures such as records and fields in a database application. A general paradigm for the representation of process plant objects and rules for reasoning about them is presented in the following sections.

The initial definition of the plant and the results of rules applied in the KBS involve the production of assertions about objects in the symbolic "world". If this process of making assertions is considered as a language of description, there must be a formal definition of that language, a syntax, in order to facilitate symbolic reasoning in the language. The structural description of a chemical plant and any rules reasoning about that plant description must talk in common terms in the language. This knowledge management is achieved by the use of knowledge representation syntaxes. These are essentially generic definitions of objects and attributes that may exist in the representation of and the rule based reasoning about a chemical process.

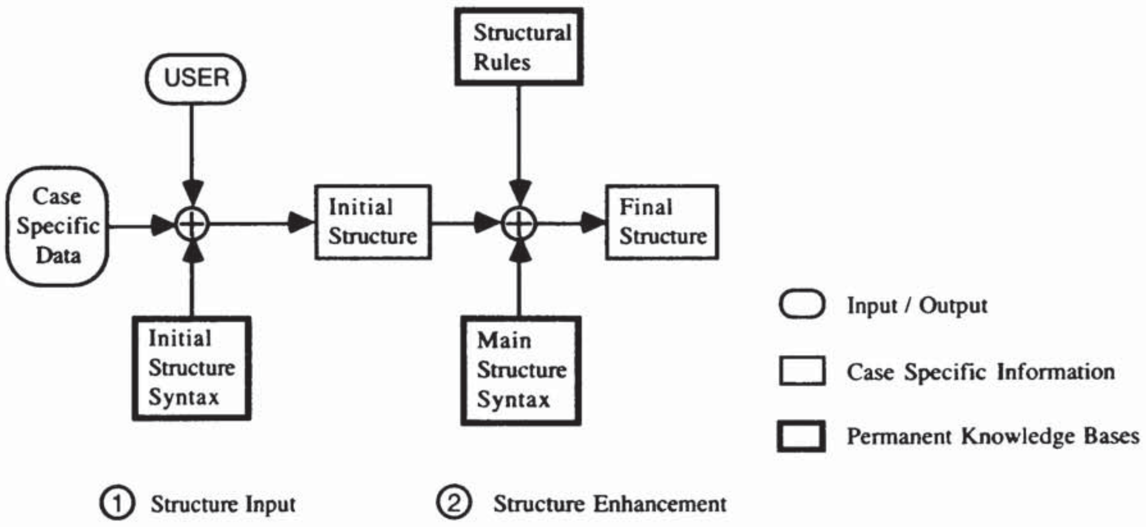
However it is one thing to be able to make assertions about objects and another to decide which particular assertions are applicable to a given object and which are not. An incorrectly defined rule may result in an invalid assertion being made about a particular object. To help prevent this, all objects which may exist in the symbolic world have generic definitions, covering all possible attributes, to which they must conform.

6.3 STRUCTURE - REPRESENTATION AND ENHANCEMENT

6.3.1 **Introduction**

In order to reason about causality in a chemical plant, it is first necessary to represent the **structure** of the plant, ie its topology and operating specifications. An overview of the way this is done is given in Figure 6.3.1a.

FIGURE 6.3.1a Overview of the Process of Structural Representation



The two main identifiable steps are:

1) Raw, case specific data, consisting of the plant topology and specifications are input by the user, assisted by and according to an initial structure syntax.

2) The initial structure is enhanced and completed by structural rules, in accordance with and up to the standard of the main structure syntax.

The following sections of this chapter describe this process of representation as follows:

6.3.2 - A general picture of the structural representation system.

6.3.3 - Describes the initial structure description (1 in Fig 6.3.1a) with respect to the ALDS Two Phase Splitter

6.3.4 - Describes the enhancement process leading to the final structure representation (2 in Fig 6.3.1a) with respect to the ALDS Two Phase Splitter.

6.3.2 **Representation of Process Plant Structure**

The representation of the process plant structure begins with a hierarchical systems decomposition identifying objects of equipment and process conditions. This is best illustrated graphically by the representation of the Ammonia Let-Down System (ALDS) Two Phase Splitter, the P&ID and the graphical representation of which are shown in Figure 6.3.2a.

As may be seen, there is a clear correspondence between the PID and the graph. The graph assists in clarifying the breakdown of the PID into systems, components and the arcs linking them. All the elements which go to make up the representation of the physical structure are referred to as **objects**.

The PID is considered to consist of a hierarchy of objects; these are unit operations, vessels, lines etc, down to the primitive components such as valves and instruments, represented on the PID by their drawing symbols. These objects are <u>systems</u>, <u>nodes</u>, <u>arcs</u>, <u>phases</u>, <u>components</u> and <u>variables</u>.

A hierarchical decomposition of a PID might be:

- Define the PID as the highest level **system**, containing:
- Unit operations as **systems**, each containing:
- Vessels and lines as **systems**, each containing:
  - **nodes**, at
    - devices (valves, instruments)
    - places of interest (junctions, inflows, levels, bulk phases)
  - **arcs** linking **nodes** (material flows, signals)
- **Phases** present at **nodes** and in **systems**, containing:

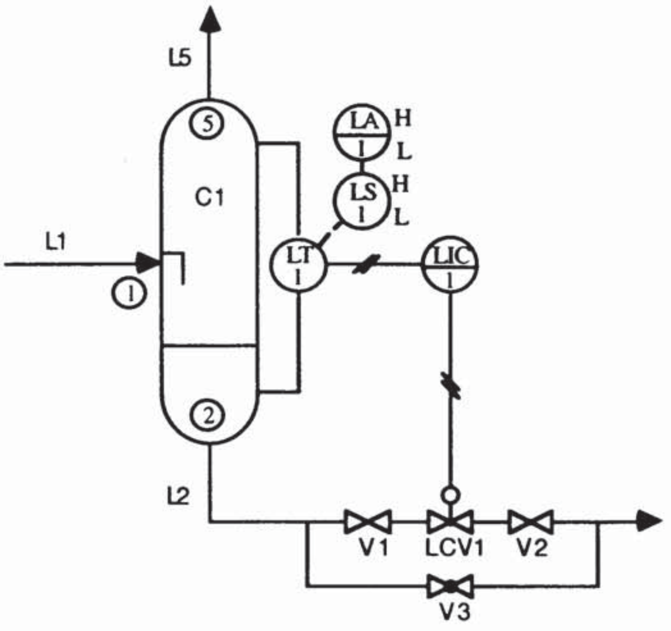
- **Compounds** present in the **phases** of the process.
- **Variables** where appropriate in the process.

The bold items here are the six types of object to be found in the Two Phase Splitter: system, node, arc, phase, compound and variable. Within each of these types of object there is a further subclassification to identify each object uniquely. This classification is illustrated in Figure 6.3.2b. Here each of the capitalised primitives at the bottom of the hierarchy identifies an object uniquely.

This hierarchy is in fact an is-a hierarchy allowing for inheritance of attributes by a given object from its 'parents' - those objects directly above it in the hierarchy. An object may inherit attributes via more than one path in the hierarchy. For example, a **CONTROL-VALVE** is-a **valve** and is-a **actuator**. A **valve** is-a **device** which is-a **node** which is-a **object**. All items in the representation of the structure of the plant are **objects**. A control-valve may have unique attributes of its own but also inherits attributes common to all its 'parents': valve device etc. Each of the elements in the hierarchy is referred to as a **type**: for example a sensor is a type of object, it may be a process-switch or another object of type transmitter.

FIGURE 6.3.2a Two Phase Splitter - PID + Graphical Representation

P+ID



Graphical Representation

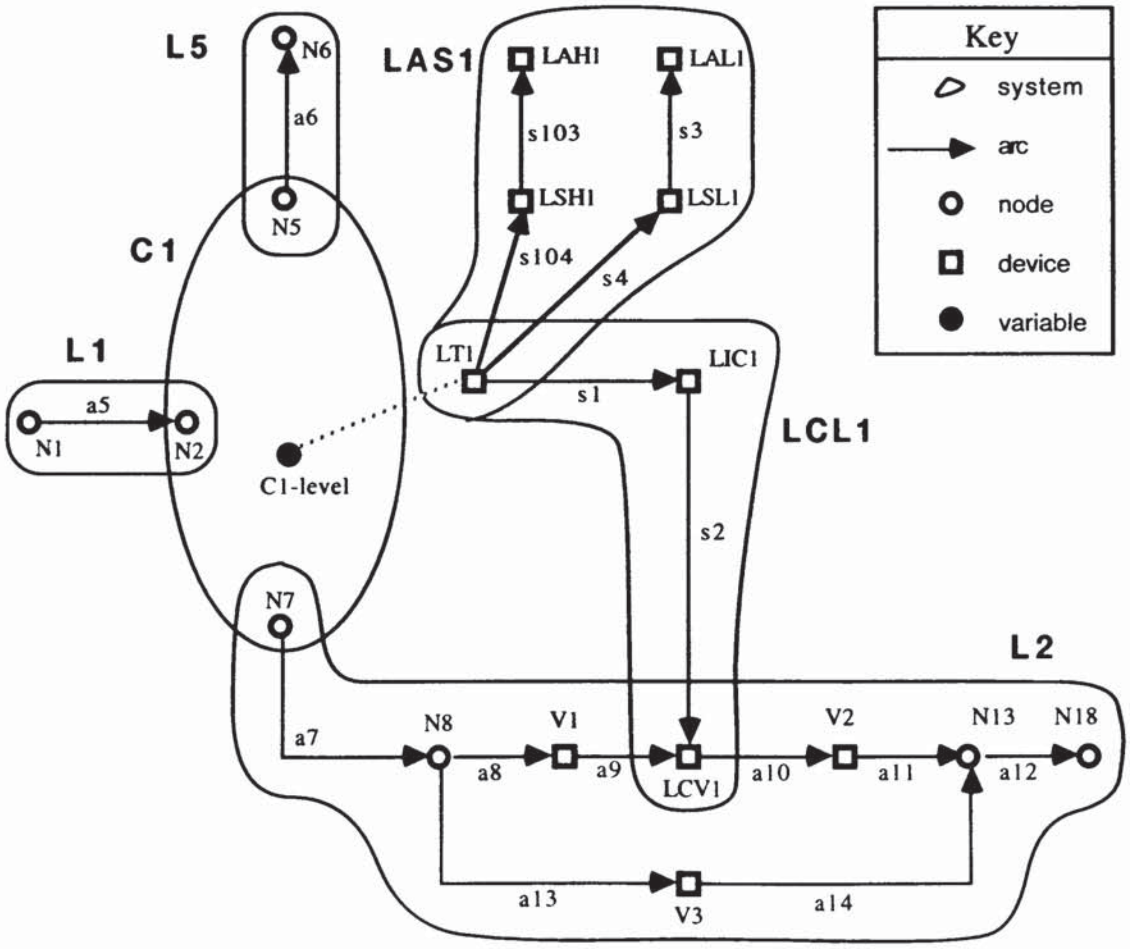
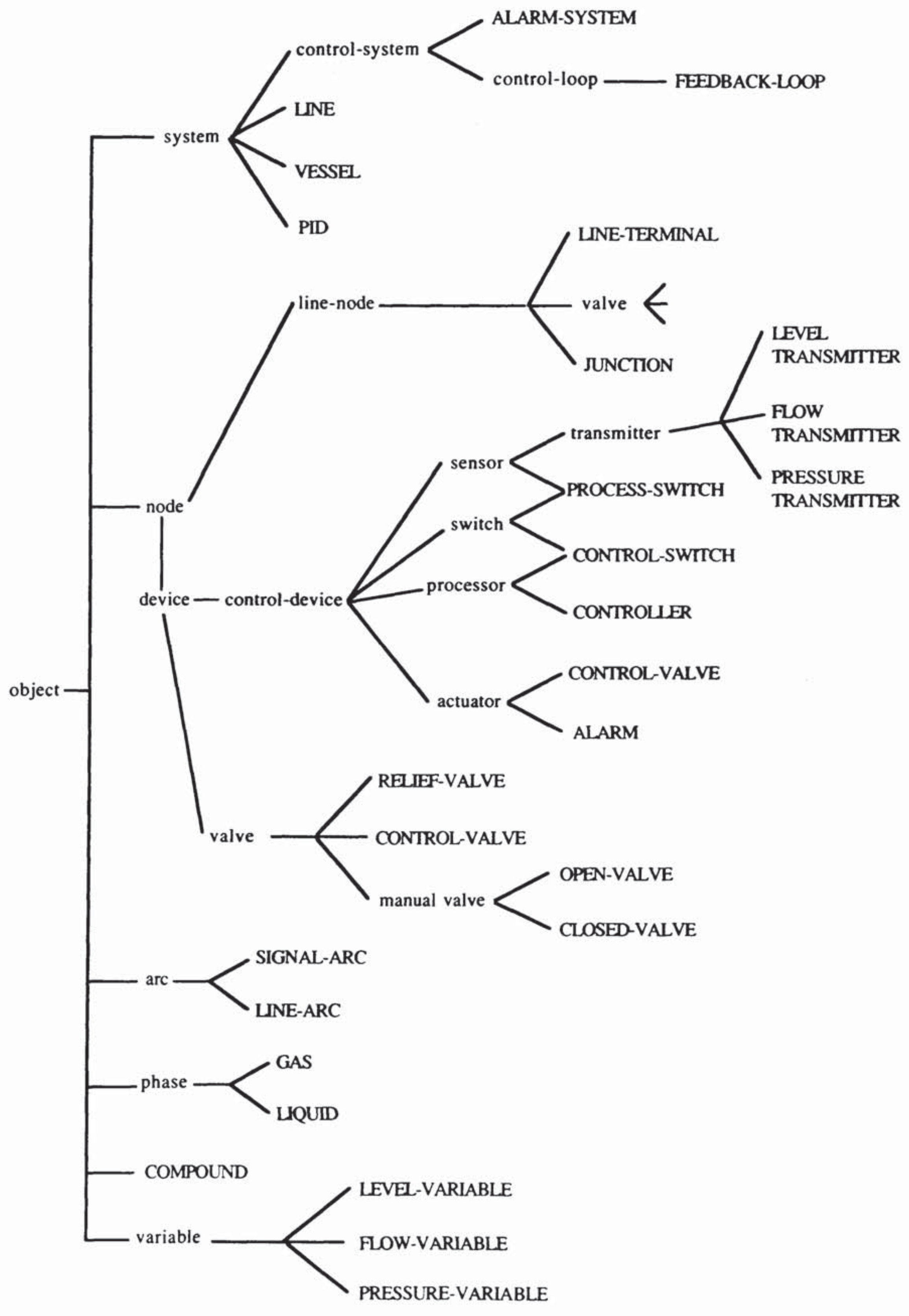


FIGURE 6.3.2b The is-a Hierarchy of Objects



Each object is defined in LISP by a frame. Figure 6.3.2c gives the frame for the control valve LCV1.

FIGURE 6.3.2c Frame for LCV1.

```
(LCV1 (instance    (:= control-valve))
      (configured  (:= air-to-open))
      (source-of    (:= a10))
      (target-of   (:= s2 a9))
      (function    (:= reduce pressure)))
```

The **attribute**s of an object are stored as **slots**, containing appropriate **value**s. In the frame for LCV1 the **instance** attribute is CONTROL-VALVE. The frame is a nested association list permitting access to attribute values via a **path**. The path to retrieve the instance slot of LCV1 is (instance :=).

The intention is to permit the definition of new paths and values, and the collection of attributable information into a hierarchy, as desired. The initial representation is kept as simple and flexible as possible to reduce any reworking which may be necessary.

It is necessary to define an individual object in two ways; leading to two types of slot:

[1] DESCRIPTOR slots

A context free description of the object in isolation, eg {configured := air-to-open}.

[2] RELATOR slots

A context specific definition of the object in terms of its relationships with other objects, eg {target-of := s2 a9}.

The links from an object to others are defined by the object's relator slots. Note that each relator has a reciprocal.

SYSTEM-OF :=

The slot (system-of (:= X Y)) in the frame for object indicates that X and Y are subsystems, nodes, or arcs contained within the system.

Similarly:

SOURCE-OF :=	A node may be the source of an arc
TARGET-OF :=	A node may be the target of an arc
FUNCTION MONITOR :=	
CONTROLS :=	Control Systems monitor and control variables.
COMPOUND :=	A phase consists of one or more compounds

Although some specific examples are cited here, apart from INSTANCE, there are, in principle, no restrictions on the descriptors and relators that may be attributed to an object by the user as long as this process is documented in the Initial and Main Syntaxes.

This method of representation provides a facility for describing objects in the plant, but how are we to know exactly which attributes to define for a given object we wish to represent? This must be done correctly and in an organised fashion so that any rules will be able to work on the plant description.

The organization of the object attributes is achieved by maintaining, separately from any individual case, a library of generic object definitions. These generic definitions are organised in an is-a hierarchy as shown in Figure 6.3.2b. However, there are two conflicting requirements for the process representation:

1) It is desirable to keep the initial description of the process by the user as simple as possible.

2) The final representation of the process must be sufficiently comprehensive to facilitate rule-based reasoning about causality within the process.

In other words, the goal is to attain a level of description which is both necessary and sufficient for the fault analysis that is to follow.

These objectives are achieved by maintaining two libraries of generic object definitions, called syntaxes, one defining the initial representation and the other defining the final representation. A rule base is used to enhance the initial representation of the plant to produce a comprehensive, final plant representation. This process is described in the next two sections.

6.3.3 **Initial Structure Representation**

There are three stages in the definition of the initial structure representation of the process:

1) Construct a graph of the plant (Figure 6.3.2a).
2) Identify all objects in the plant .
3) Define the attributes of each object in the plant according to the initial syntax.

The most important step to get right here is the identification of all objects present. Once this is done the initial syntax ensures the presence of the required attributes and therefore that the initial representation of the process is sufficiently comprehensive. A list of the initial objects present in the Two Phase Splitter representation is given in Figure 6.3.3a.

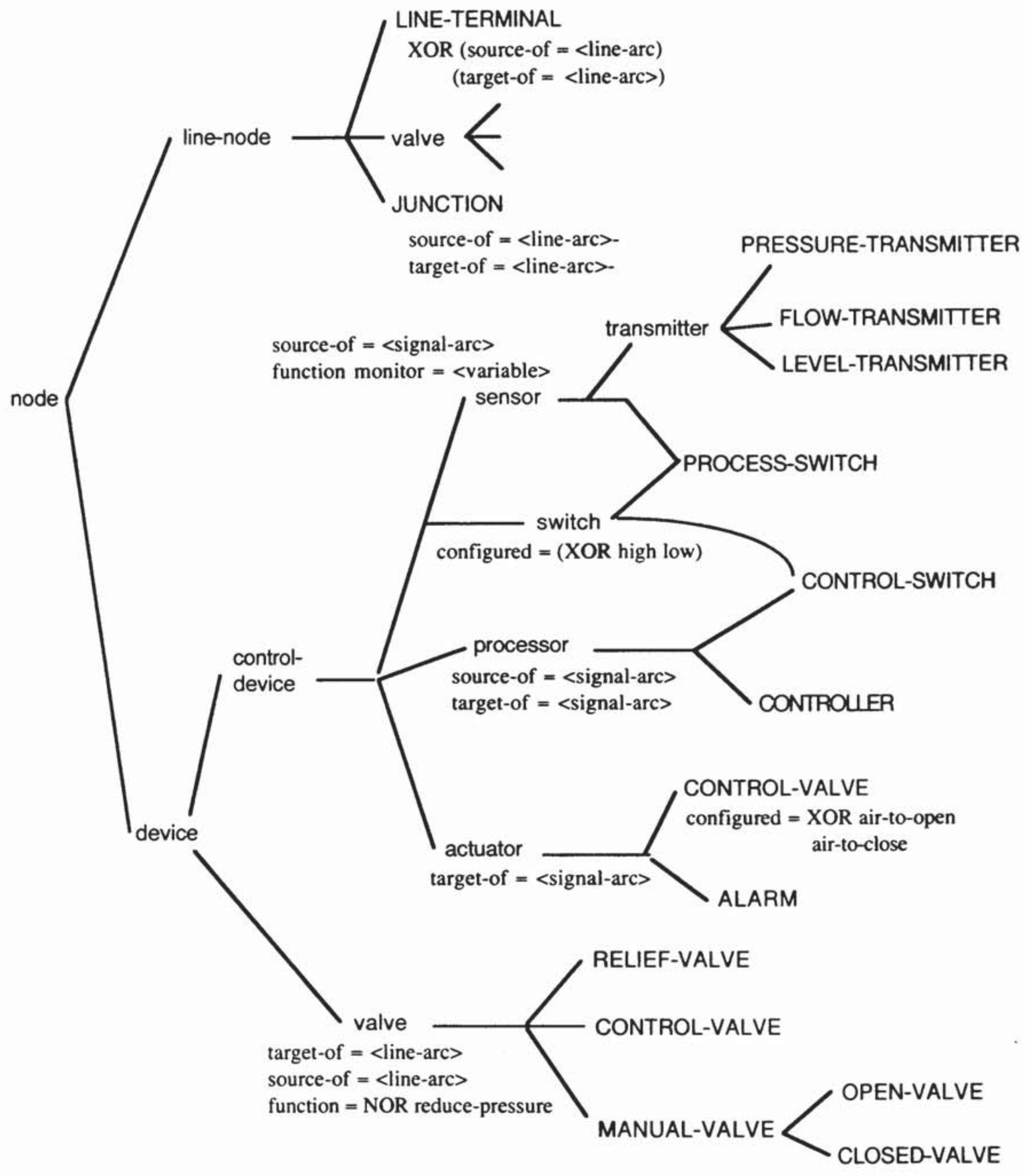
FIGURE 6.3.3a Two Phase Splitter - Initial Objects

```
system
    PID            ALDSPID
    LINE           L1 L2 L5
    VESSEL         C1
    FEEDBACK-LOOP  LCL1
    ALARM-SYSTEM   LAS1
device
    LEVEL-XMITTER  LT1
    CONTROL-SWITCH    LSH1 LSL1
    CONTROLLER     LIC1
    CONTROL-VALVE  LCV1
    ALARM          LAH1 LAL1
    OPEN-VALVE     V1 V2
    CLOSED-VALVE   V3
line-node
    LINE-TERMINAL  n1 n2 n5 n6 n7 n18
    JUNCTION          n8 n13
arc
    SIGNAL-ARC     s1 s2 s3 s4 s103 s104
    LINE-ARC       a5 a6 a7 a8 a9 a10 a11 a12 a13 a14
phase
    LIQUID         NH3-syngas
    GAS             syngas-NH3-CH4
compound
    NH3 CH4 syngas
variable
    LEVEL-VARIABLE    C1-level
```

The initial syntax consists of an is-a hierarchy of generic objects. Figure 6.3.2b outlines the entire is-a hierarchy, Figure 6.3.3b gives the initial syntax for nodes. The hierarchy represents a set of generic, stereotype frames which are instantiated to represent objects in the plant in question. The stereotype frame for a given object is defined by collecting slots inherited by the object in the hierarchy.

As may be seen, the Initial Syntax defines relations between objects, this implies that if we know that one object exists, eg an actuator, then there must also exist a signal arc such that {actuator target-of := signal-arc}, cf Figure 6.3.3b. Similarly that signal arc will have a source, which may be joined to other arcs. It follows from this that if a single object is specified, the Initial Syntax will ensure the inclusion of others. These others will be related to more objects, and in this way all related objects will be defined. It may be possible, however, for some independent objects to "slip through the net" of the initial syntax. As the enhancement of the representation continues towards the Main Syntax, as in the next section, any missing items will be picked up as undefined and the user may be queried to complete the list of objects. This satisfies the key requirement here, as in (2) above, that there be a complete list of objects present.

FIGURE 6.3.3b Initial Syntax for Nodes



The attributes - slots and slot values - that a given generic object may have are shown in the initial syntax. Allowed slot value formulae conform to a syntax as shown in Table 6.3.3a.

TABLE 6.3.3a Stereotype Slot Values

Value Formula	**Value Permitted**
symbol	- the symbol is the only value possible
<object>	- another object in the hierarchy
<object>-	- one or more objects of the defined type
AND x y ...	- all of the symbols, types or formulae x y ..
OR x y ...	- any one or more of the symbols or objects or formulae x y ...
XOR x y ...	- any one of the symbols, objects or formulae x y ...
NOR x y ...	- NIL (empty) or any one of the symbols, objects or formulae x y ...

The initial stereotype frame for a control valve and its instantiation as LCV1 in the Two Phase Splitter are shown in Figure 6.3.3c

FIGURE 6.3.3c Initial Stereotype and Instantiation for LCV1

control-valve		**LCV1**	
configured	:= XOR air-to-open air-to-close	instance	:= control-valve
target-of	:= <signal-arc> <line-arc>	configured	:= air-to-open
source-of	:= <line-arc>	target-of	:= s2 a9
function	:= NOR reduce-pressure	source-of	:= a10
		function	:= reduce-pressure

Once the initial structure representation is complete, then it is possible to reason about it by the application of rules. The first step is reasoning about the structure of the plant; this is described in the next section.

6.3.4 **Final Structure Representation**

6.3.4.1 Overview

This section describes the transformation of the initial structure representation into the final structure representation. This transformation is achieved by the application of structural rules. These rules are compared against the structural representation and, if a match is found, new assertions are added to the representation. These assertions take the form of new attributes for objects or may involve the creation of new objects. The end result of the application of structure rules must conform to the main structure syntax; this is similar to the initial syntax but has more attributes for the objects and defines all the necessary and sufficient slots and permissible values for those slots which are required for further reasoning.

6.3.4.2 Structural Rules

These rules take the general form: IF <pattern> THEN <action>:

<pattern> consists of a pattern of objects and attributes to be compared to the structure representation.

<action> consists of a set of assertions about objects in the structure, which, in the context of any matched pattern objects, involves either the attribution of slot-value combinations or the creation of new objects.

Perhaps the simplest rule is the one that defines the reciprocal of an already existing relation between two objects:

IF object1 source-of := object2 THEN object2 has-source := object1

If this rule is compared against the initial structure representation for the Two Phase Splitter, one of the matches will give:

object1 = LCV1   object2 = a10

leading to the definition of the attribute: a10 has-source := LCV1.

There are three basic types of structural rule:

1) General Structure Rules

These rules:

- complete reciprocal relators as above
- complete the hierarchical decomposition of the structure
- define more abstract relators such as upstream and downstream relations, and whether lines are inflows or outflows of vessels
- define the presence or absence of phases and compounds

One example of the purpose of these rules is illustrated by the last category. In order to specify events involving contamination, {flow as well as} it is necessary to pin down all phases and chemical species in the process representation.

2) Flow Rules

These rules define the flow or no-flow situation which may exist in lines or vessels, deducing the situation from the structural representation of the process and the presence or absence of phases. It is necessary to know the flow situation at, for example, a valve, to determine the applicable failure mode for that valve. If the valve is normally open and there is a flow present, then {valve closed} is an applicable failure mode but {valve open} is not. If there is no flow present then any failure mode will only be an enabling failure. An example of the simplest flow rule is:

IF {v is-a := valve} AND {v status := closed} THEN {v flow := no}

The general philosophy behind line flow rules is:

1) Decide on no-flow situations (at closed valves for example) and propagate this no-flow situation in the relevant parts of lines.
2) Where no no-flow situation can be proved and a phase is present, assume {flow := yes}

This process presents a conflict resolution problem which is solved by procedural means.

3) Function Rules

These rules reason about the function of objects in the structural representation and may involve the creation of new variables where appropriate. For example:

- A Control System has monitored, manipulated and controlled variables.
- A variable may therefore be defined as monitored, manipulated or controlled.
- At a control valve, a new variable is created; this becomes a manipulated variable of the control system to which the control valve belongs.

6.3.4.3 The Main Structure Syntax

The main structure syntax, like the initial syntax, consists of a generic is-a hierarchy of attributes which represents generic frames instantiated as objects in the structural representation. In fact the initial syntax is a subset of the main syntax. The main syntax defines all the attributes that an object may have in the final structure representation. An example, the final representation of the control valve LCV1, is given in Figure 6.3.4a.

FIGURE 6.3.4a Final Frame for LCV1

```
(LCV1
    (instance (:= control-valve))
    (configured (:= air-to-open))
    (target-of (:= s2 a9))
    (source-of (:= a10))
    (function (:= reduce-pressure))
            (manipulate (:= LCV1-flow)))
    (has-system (:= L2 LCL1))
    (joined-to (:= s2 a9 a10))
    (status (:= open))
    (upstream-of (:= V2 n13 n18))
    (downstream-of (:= C1 n7 n8 V1))
```

Once the final representation of a particular process is complete, it is possible to consider events and their propagation within the process. This is discussed in the following sections.

6.4 EVENTS - REPRESENTATION AND DEFINITION

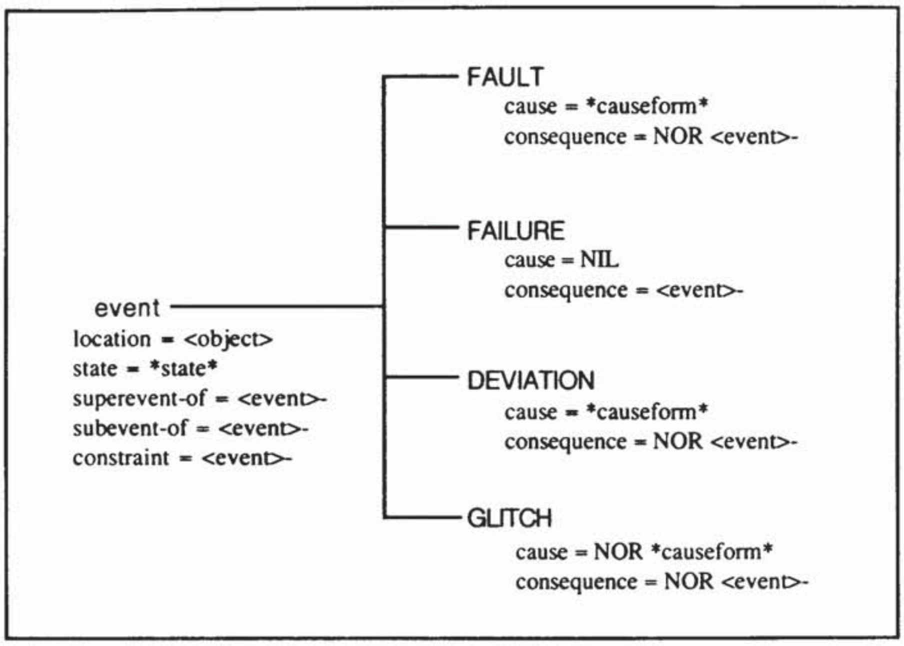
6.4.1 Introduction

Generally speaking, a fault analyst is only interested in those events he considers to be "significant". For the study of detailed fault propagation it is necessary to consider events at a much greater level of detail, although these events will not appear in the final fault analysis. This section describes the different types of events in the proposed design, how they are represented, and how they are defined for a given case study.

6.4.2 Representation of Events

In the global scheme of the knowledge representation, the highest type definition is **entity**. One type of entity is an **object**, physically present in the chemical plant structure, another type is an **event**, Figure 6.4.2a. Thus, to the KBS, every "thing" that is known about in the knowledge bases is of type **entity** and may be included in a syntax, facilitating knowledge management.

FIGURE 6.4.2a Event Types and Syntax



There are four types of event:

FAILURE:	This is a primary failure mode of a device, deemed to have no causes, but must have consequences.
FAULT:	This is a command fault occuring at a device, and must have causes and may have consequences.
DEVIATION:	This is a process deviation which is a command fault kind of event occurring anywhere apart from a device and must have causes and may have consequences.
GLITCH:	This term covers those events which are generated by the KBS as part of detailed reasoning about causality. Glitches do not appear in any final fault analysis.

Failures, Faults and Deviations are the events that may be defined in the valid event set and that will appear in the final fault analysis. The Glitches represent a more detailed reasoning about causal propagation which are included where appropriate.

As may be seen from Figure 6.4.2a, an event is defined in isolation, ie context-free, by two attributes:

location := <object>

- the location is the physical object where the event occurs.

state := *state*

- the state that the object enters into when the event occurs. *state* indicates a device fault or failure, or a variation in a plant parameter such as temperature, given as a complex and unique value set which may here be syntactically confirmed using a structure representing the Table of Guide Word, Property Word and Component, as in Table 2.4.5b.

```
Thus an event might be defined as:     (event1   (instance (:= fault))
                                                 (location (:= LCV1))
                                                 (state (:= open more)))
```

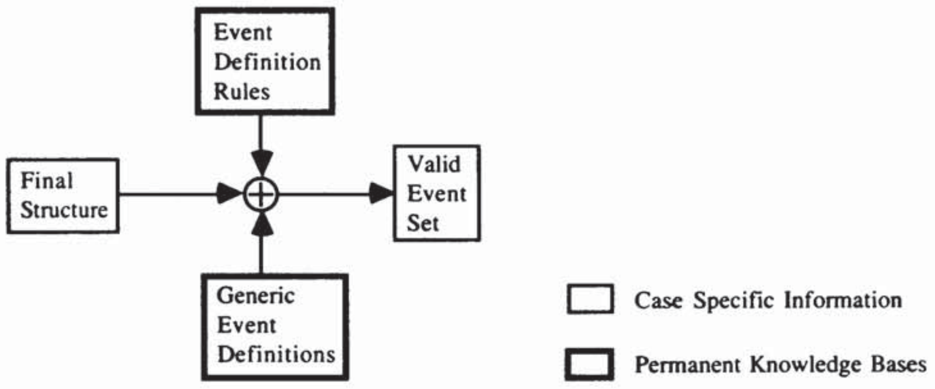
The symbol 'event1' is called the **identifier** of the event and is an internal symbol generated and maintained by the KBS in order to uniquely identify the frame representation of the event. This is transparent to the user as the event appears as {LCV1 open more}.

6.4.3 **Definition of Events**

6.4.3.1 Introduction

At this stage the representation of the structure of the process is complete and the next step is to determine those events of significance: failures, faults and deviations which may occur in the process. This is done in two ways, as shown in Figure 6.4.3a, taken from Figure 6.2.a.

FIGURE 6.4.3a Definition of Events from Structure Representation

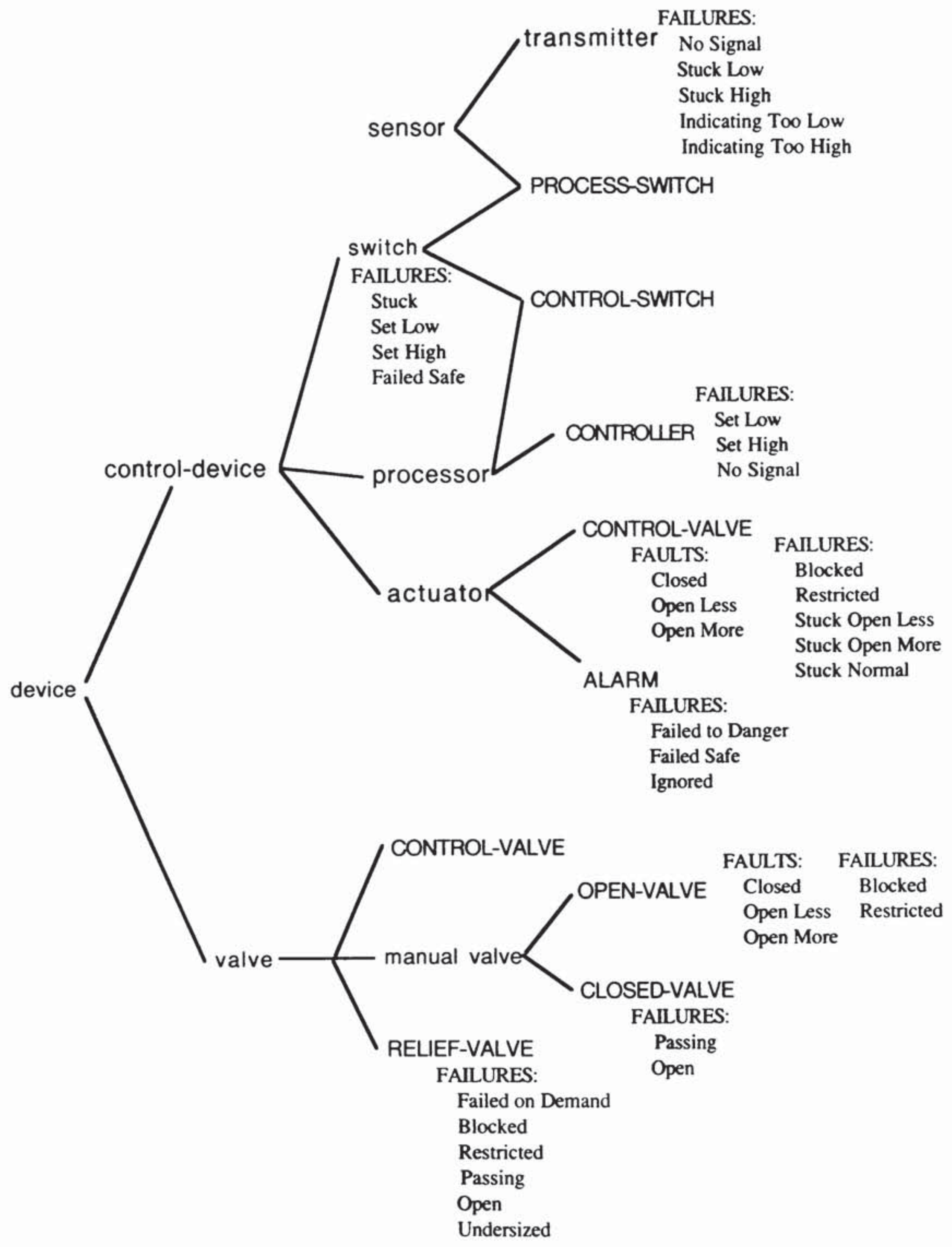

Event Definition Rules
Final Structure
Valid Event Set
Generic Event Definitions
Case Specific Information
Permanent Knowledge Bases

There are two kinds of event that require definition:

1) Those events which are context free, ie they can occur at a particular object regardless of the particular structure in which the object exists. Examples are device failure modes. These events are determined by GENERIC EVENT DEFINITIONS.

2) Those events which are context dependent; in order to determine if they exist, the context of the object which is the location of the event must be taken into account. Examples include line flow deviations. These events are determined by EVENT DEFINITION RULES.

FIGURE 6.4.3b Generic Event Definitions for Devices



6.4.3.2 Generic Event Definitions

From an analysis of the events defined in the ALDS case study, and with reference to the Failure Mode Tables given in Appendix 1, it is possible to identify those context-independent Faults and Failures which are applicable to devices in the Two Phase Splitter. These generic event definitions are maintained in a similar fashion to the structure syntax, by means of an is-a hierarchy, as shown in Figure 6.4.3b for devices.

Placement of these events in the context of the representation of a particular plant is simply a matter of comparing the instantiation of objects as a particular type and attributing to the objects the relevant events.

6.4.3.3 Event Definition Rules

In some cases it is only possible to decide on the existence of an event at an object by considering the context of the object in the representation. This is achieved by using rules. A rule has a pattern to compare against the representation and, if a match is found, events may be defined corresponding to that match. An example of a simple event definition rule and its application to the Two Phase Splitter is given in Figure 6.4.3c.

FIGURE 6.4.3c Example Event Definition Rule

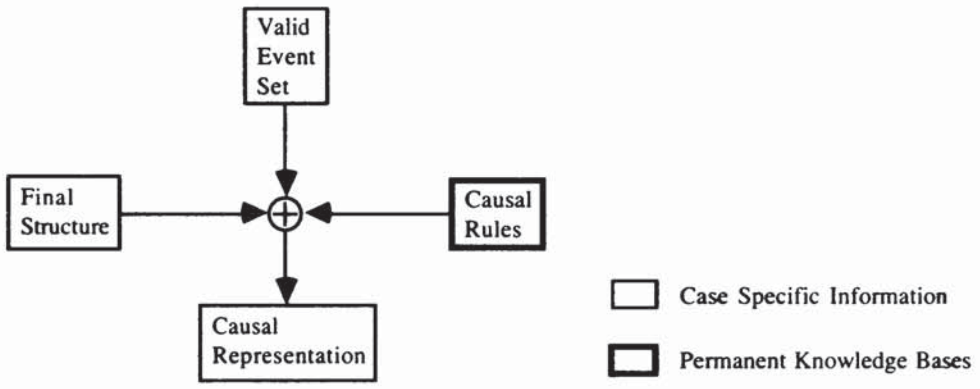
```
IF  v  is-a := vessel               THEN e  instance := deviation
    lv  is-a := level-variable           e  location := v
    v  system-of := lv                   e  state := level no
                                                     level less
                                                     level more

This Matches:       v = C1               lv = C1-level

Giving events e =   C1 level no     C1 level less    C1 level more
```

6.5 CAUSALITY - REPRESENTATION AND REASONING

6.5.1 Introduction

To set the stage for the explanation of causal reasoning it is useful to consider the state of the Knowledge Based System at this point, Figure 6.5.1a.

FIGURE 6.5.1a Causal Definition in the KBS



Here the final structure representation of the process is available, and a complete set of the events which will appear in the causal analysis has been constructed. The next step is to develop causal links between these events; this is done by the application of causal rules. The representation system used for describing causality in the KBS and the application of causal rules to determine fault propagation characteristics are discussed in the next two sections.

6.5.2 **The Representation of Causality**

As may be seen from the event syntax in Figure 6.4.2a, an event is defined in isolation, ie context-free, by its location and state attributes. The representation of causal relations between events uses three attribute types.

1) cause := *causeform*

   Here the *causeform* is any Boolean form involving the logical connectives {AND OR NOT} and has event identifiers as primitives. An example is {e1 cause := (AND e2 (OR e3 e4))}, where e1, e2, e3 and e4 are event identifiers.

2) consequence := <event>-

   A consequence of an event may be one or more other events, eg {e1 consequence := e2 e3}

3) Event Sets

   The two attributes {superevent-of := <event>-} and {subevent-of := <event>-} are logical reciprocals, ie {e1 superevent-of := e2} and {e2 subevent-of := e1} are two ways of expressing the same relation. This relationship is used when there is no clear causal relation between two events and yet they are clearly linked. A good example is a flow deviation in a line and a similar flow deviation at a point in the line, eg {L2 flow no} and {LCV1 flow no}. These two events may be considered to be, in actuality, different ways of expressing the same plant state; if either of the two occurs then the other has also occurred. This cloudy causal definition is resolved by considering the first event {L2 flow no} as an event-set, containing set members {LCV1 flow no} and the other no-flow deviations in the line L2. Thus the first event is a superevent of the second. In the final fault analysis the second event may be said to cause the first, as it represents an internal failure of the line. (A potential problem of this approach is that a breach in a line may result in no flow in part of the line, and continuing flow upstream of the breach. This may perhaps be resolved because from the point of view of fault propagation downstream, the conclusion no flow in the line is valid).

This concept of an event set corresponds to one of the guidelines for Fault Tree Synthesis that are discussed, amongst others, in section 2.5.3: causality never passes through an OR gate [Haasl 1981]. Inputs to an OR gate in a Fault Tree are identical to the output but are more specifically defined as to cause. {LCV1 no flow} is identical to {L2 no flow} but is more specifically defined as to cause.

6.5.3 **The Definition of Causal Relationships.**

Causal relationships between events are defined by the application of CAUSAL RULES. The general form of a causal rule is: IF <condition pattern> THEN <action-pattern>.

<condition-pattern> is a pattern of assertions about entities (objects and events) and relations between entities to be compared against the case specific knowledge: the final structure, valid event set and the causal representation that is generated by the previous application of any causal rules.

<action-pattern> is a pattern of assertions about events. These may apply to events in the valid event set or may involve the creation and attribution of GLITCHES - events too detailed to be of interest in the final causal analysis but useful in representing causal propagation.

Causal rules may operate not only on the structure and event set but also on the results of previous application of causal rules - the causal representation. In terms of the Knowledge Based System the distinction between the valid event set and the causal representation in Figure 6.5.1a is a conceptual one; in practice they are combined into the same data structure.

As may be seen from the previous section it is possible to define causality between events using the cause, consequence and superevent/subevent relators.

It is possible to divide the causal relators into two groups, those which directly link significant events (faults, failures and deviations) and those which involve the generation of glitches. However this is a conceptual division and the rules are in fact grouped according to the type of objects on which they operate. This grouping is effected in order to manage the knowledge base of causal rules. The grouping of rules is as follows:

Devices

These rules determine how the flow of causality from the inputs to the outputs of a device is affected by the failure modes and fault of the device. For example, the state of a transmitter, failed or operating normally, influences the relationship between the variable it monitors and the signal that it transmits. These rules may be thought of as representing mini-fault trees for devices.

In the ALDS case study, a large number of rules were developed for device failures as it is appropriate to consider all the failure modes of a device at one time, ensuring consistency, rather than just those directly applicable to the ALDS.

Transition tables, decision tables and mini-fault trees from the literature were a valuable source of information in formulating these rules.

Vessels

These rules describe how causality propagates through vessels, for example temperature and flow deviations from inlets to outlets. The assumption is made here that "all other things are equal". In this particular case, only a few rules were necessary, applicable directly to the ALDS vessel C1.

Lines

These rules describe event propagation within lines. Flow deviations may be considered to propagate upstream and downstream from the point of initiation. Line deviations may be considered to be superevents of node deviations. Phase and Compound deviations are also considered. A considerable number of general rules of this type were developed, as they were seen to be applicable to many different flow conditions.

General

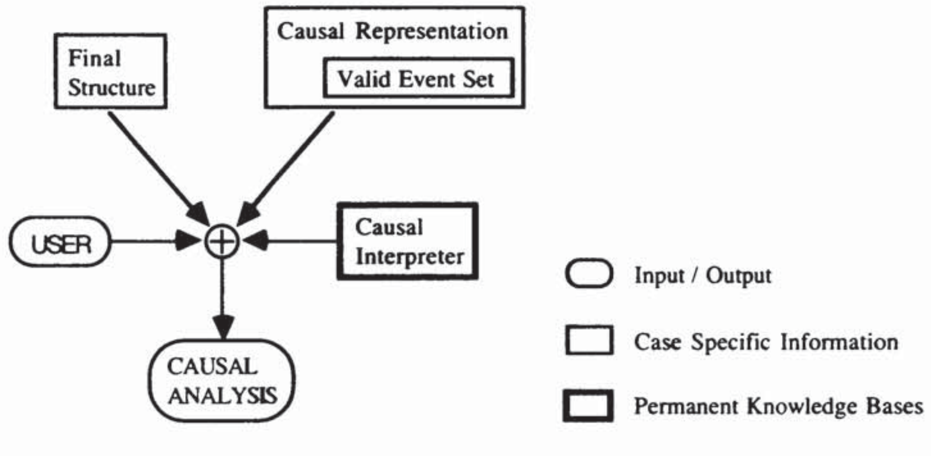
This category contains those few rules that do not fit into any of the above groups.

6.6 PRODUCTION OF THE FAULT ANALYSIS

6.6.1 Introduction

Production of the fault analysis involves the user's interaction with the Structure Representation and the Causal Representation which contains the Valid Event Set. This process is conducted using a Causal Interpreter, Figure 6.6a.

FIGURE 6.6a Production of the Causal Analysis



The Causal Interpreter provides a procedural means of eliciting the required analysis from the Causal Representation and Final Structure. In this case the goal is primarily to produce Cause and Symptom Equations (CSEs).

6.6.2 HAZOP using Cause and Symptom Equations

The first step in the production of CSEs is the selection of those events for which we wish to develop the causes and consequences. These events exist in the previously defined Valid Event Set. This selection of events is done in a similar manner to that described for HAZOP in section 2.4.3 and illustrated in Figure 2.4.3a. The procedure used here is shown in Figure 6.6.2a. Symptom equations are used to describe the effects of process deviations at the inlets to a vessel, on the outlets of the vessel.

FIGURE 6.6.2a Production of Cause and Symptom Equations

```
FOR each major Vessel
        FOR the vessel   →
            each line    →
             each auxiliary → FOR all guide words
                                    FOR all credible deviations in Valid Event Set
                                                  (from Guide Word)
                                              $\boxed{\text{Develop Cause Equations}}$
                                    ENDFOR
                              ENDFOR
        ENDFOR
        FOR  all events at inlets to the Vessel
                $\boxed{\text{Develop Symptom Equations}}$
        ENDFOR
ENDFOR
```

There are two further procedures in Figure 6.6.2a which require description. These are the development of Cause Equations and the Development of Symptom Equations.

Developing Cause Equations

The event for which the Cause Equation is to be developed is known, and is identified in the Causal Representation. The Cause Equation comes about by a process similar to a mini-fault tree analysis. The selected event is considered as an effect in the causal representation and any causes of this event are found in the cause, superevent or constraint slots of this event. If a cause is a Glitch it is explored for causes of its own. The process of backward causal tracing continues until the primaries of this mini-fault tree are all significant events (ie faults, failures or deviations), or events at the boundaries of the system. The Cause Equation is then complete. This system generated Cause Equation may then be examined by the user who may make corrections to it or enquire how it came about.

Developing Symptom Equations

Symptom Equations are produced in a similar manner to Cause Equations but in this case propagating forwards - from cause to effect - through the Causal Representation from the selected event, perhaps via Glitches, to consequential significant events, ie Faults or Deviations.

6.6.3 Fault Tree Synthesis

The first step in a fault tree synthesis is the selection of a suitable top event. This may be, for example, a hazard presented by the release of a particular material. The next step is the breakdown of the top event into specific events which occur at plant level, eg releases at specific locations; these events are of a class which are known to the expert system and which may be developed further.

As may be seen from the above description, the process of generating Cause Equations is essentially that of Fault Tree Synthesis. Here one of the identified plant level top events is selected from the Valid Event Set and its causes explored, through the Causal Representation, via Faults and Deviations towards primary events. This process of successive refinement of the Fault Tree may be guided by the user at each step.

By definition, Faults and Deviations must have causes and therefore must be developed further in the fault tree. This fact also provides a useful guide to any shortcomings of the expert system, in cases where no further causes of a Fault or Deviation are found. This may lead to the amendment of existing rules and structures or the creation of new ones.

6.6.4 Other Fault Analyses

6.6.4.1 Tabular Hazop

In a tabular HAZOP the expert system would be required to assist in eliciting all causes and consequences of events derived from application of Guide Words. These events will be given by the set of Deviations (as defined in the expert system) contained within the Valid Event Set. The structured listing of the Deviations may be produced using the Lihou algorithm shown in Figure 6.6.2a, which covers all vessels, lines and auxiliaries in the P&ID.

For each deviation it is necessary to find the ultimate causes, given by Failures or boundary events, and ultimate consequences, given by other Deviations, Faults or boundary events which have no apparent further consequences. This may be done by tracing through the causal network in a "backward" and "forward" causal manner in a way similar to that described for cause and symptom equations.

6.6.4.2 Failure Modes and Effects Analysis.

In an FMEA, a particular item of equipment is selected and an exploration of the consequences of each of its failure modes is made. After selection of the item, its failure modes may be derived from the Valid Event Set as those failures which relate to that item. The effects of each failure mode may be found by tracing causally forward through the causal network to ultimate consequences.

CHAPTER SEVEN

DISCUSSION

7.1 INTRODUCTION

The previous chapters described the proposed design for a Knowledge Based System and how that design came about.

This Chapter addresses two main points. The first discusses the key features of the design and assesses the design with regard to the Case Study. The second considers the generalization of the knowledge bases to cope with other domain problems and describes how an operational KBS might be constructed based on the principles arising from this research.

7.2 DESIGN EVALUATION

7.2.1 **Introduction**

The suitability of the proposed design for a KBS for Fault Analysis rests on two attributes of the design:

1) The Architecture

This is the overall structure of the design, facilities for creating, maintaining and processing knowledge bases, and facilities for communicating with the system builder and end user. The architecture may be viewed as the Knowledge Based System without the Knowledge, ie a "shell".

2) <u>The Knowledge</u>

The strength of a KBS is naturally dependent on the quality of the knowledge contained therein. In this case this refers to the contents of the knowledge bases:

- Initial Structure Syntax and input procedures.
- Structural Rules.
- Main Structure Syntax.
- Event Definition Rules
- Generic Event Definitions
- Causal Definition Rules
- Fault Analysis construction Heuristics and procedures - dependent on the particular fault analysis desired.

In developing the proposed design the architecture was intended to be a general one, suitable for tackling a wide range of problems in the domain. The knowledge contained within the knowledge bases, however, was constructed specifically with regard to producing a Fault Analysis for the Ammonia Let Down System (ALDS).

7.2.2 **The Architecture**

The proposed KBS architecture came about through an iterative process of design, experimentation and evaluation. This process was characterised by a gradual refinement of the initial conception of the problem - mapping structure to causal space - in an attempt to simplify the overall problem by considering it as a number of separate sub-problems. There are a number of key decisions that were made in the development of the final architecture:

1) <u>Syntaxes</u>

Syntaxes are necessary for the management of complex knowledge. These syntaxes establish both the form and the content of information permitted in the knowledge bases.

i. Form

It is clearly necessary to formally define the syntax of expressions in the Facts Database and Rule Bases. This is implicit in the syntaxes, eg the syntax for the representation of events.

ii. Content

The representation of the process plant structure is, by necessity, a complex one. It is therefore desirable to have a permanent record of the types of objects that may be represented and the generic attributes which may be associated with those objects. This is achieved by the use of Structure Syntaxes, enabling a systematic means of data input and knowledge expression - new rules may be written in conjunction with the syntaxes and may involve additions to and modifications of the syntaxes.

2) Structure Enhancement

It is desirable that the data input by the user for a particular case is kept as simple as possible. It is necessary, however, for the structural description on which the causal rules operate to be as comprehensive as possible. The architecture provides for the application of Structural Rules to facilitate enhancement and abstraction of the initial description of the process.

3) The Separation of Events from Causation

The decision to separate the generation of events from the generation of causation was a very difficult one to make. On the surface, this method involves the duplication of information and appears to involve extra complexity. For example, to decide that $A \rightarrow B$ in the Fault Analysis requires first that events A and B are established, and second that A leads to B. Events A and B are therefore defined twice.

The principal justification for the differentiation of event and causal definition is to provide a **robust** KBS:

- It is considerably easier to define the events that may occur within a plant than to define causal propagation. This definition may achieve a much higher degree of completion.

- The Causal Rules may be written with a considerable degree of freedom and may express general cause effect relationships or constraints. Any spurious events or causal links that are generated will not necessarily lead to incorrect fault analysis support as the key elements of the Causal Representation are those events defined in the Valid Event Set.
- There are two knowledge bases that establish the existence of events. This means that any event missing from the application of one of these knowledge bases may be picked up by the other (although the potential for common omission does, of course, exist).

4) The Causal Representation

For reasons which are discussed extensively in Chapter 5, the proposed design involves the synthesis of a Causal Representation from which a Fault Analysis may be produced, rather than the deterministic, direct production of the Fault Analysis. This liberates the synthesis of cause-effect from the specific constraints of any particular fault analysis method. It also permits the production of multiple fault analyses from a single analysis of the plant.

5)

One of the initial goals of the research was to investigate the possibility of the application of heuristic rules to a description of the structure of the process to generate a causal analysis. These heuristic rules are, however, too general or "high level" to be applied directly. The results of this research may be viewed as the production of detailed structural and causal representations of a plant to which these heuristics may be applied. In other words, the causal network provides the foundation of "low level" knowledge on which the fault analysis is based.

7.2.3 **The Knowledge**

A principal result of this research has been the identification of the particular types of knowledge that are required to conduct a Fault Analysis. For robustness a considerable depth of knowledge is needed, operating at both a macro and a micro scale in the consideration of causation.

The particular knowledge bases constructed for the Case Study were obviously intended to solve this problem as well as possible, and it is difficult to assess, from this knowledge, the success of future application to other cases. This is discussed in section 7.3.

7.2.4 The Mechanics of Experimentation

Generally speaking, when researching into the design of a computer system for a particular problem, one is freed from the necessity of building the parts of the system that interface with the user. This means that efforts can be concentrated on writing code to solve the core problem. In developing KBS however, the success of the system depends largely on the knowledge input by the system builder: the Knowledge Engineer. During this knowledge input phase, the builder acts as a user, ie it is necessary to have comprehensive i/o facilities for the building phase. For complex problems such as the one at hand, this requires the use of a sophisticated KBS building environment using a suitable workstation, as discussed in Section 3.4.

The KBS building environment for this research rested on the use of COMMON LISP on an IBM PC/AT. For this reason only limited experimentation was feasible. With reference to the overview of the KBS architecture (Figure 6.2a) it has been possible to test the stages from the Initial Structure Representation to the production of the Causal Representation, using the Two Phase Splitter Case Study. The input of Case Specific Data and the Production of the Causal Analysis have been done largely by hand. Only a brief discussion of the use of Heuristic Rules is included. These experiments stretched the capabilities of the PC/AT to their limits.

7.2.4.1 Representation

Representation of Plant Structure

The objects which make up the plant are represented by Frames. This is readily done in LISP. Figure 7.2.4a gives the LISP symbol-structure representing the initial Frame for the Level Transmitter LT1. As may be seen, this straightforward nested list is a natural way of formally representing a Frame.

FIGURE 7.2.4a Frame for LT1

```
(LT1 (instance      (= level-transmitter))
     (source-of     (= s1 s4 s104))
     (function      (monitor (= C1-level))))
```

Access functions to define, change or retrieve attributes are easily written in LISP, using the following definition of the structure of an assertion in the Frame. (A Frame is equivalent to a set of derived assertions - Chapter 3).

<assertion> → <object> <path> = <value>

This definition is also used in pattern matching.

The Representation of Events

The representation system used for events is structurally identical to that for objects. All entities "known" have the same Frame type representation, although some Frames may involve special slot value forms. The intent was to enable a single rule processor to handle all Facts and Rules.

All entity representations are derived from or governed by syntaxes. This is done in order to ensure the correct description of entities and to provide a framework for a coherent system of representation.

7.2.4.2 <u>Rules</u>

<u>Rule Forms</u>

The general form of a rule is shown in Figure 7.2.4b

FIGURE 7.2.4b <u>General Form of a Rule</u>

```
(rulename          (pattern
                   (descriptors  (-----) . . .)
                   (relators     (-----) . . .))
      (action
                      (assert      (-----) . . .)))
```

This form bears a striking resemblance to the Frame structure in LISP. This is not surprising given that, unless it is "pretty printed" as above, every LISP expression looks very much the same. It is the simplicity of syntax that provides this uniformity.

The rule may be read as: IF <pattern> THEN <action>. The <pattern> is a set of assertions about objects to be matched to the appropriate Facts, ie Structure Description, Valid Event Set or Causal Representation. This leads to instantiation of local variables in the rules which may be used to execute the assertions of the rule and add to the Facts Database.

<u>Pattern Matching</u>

The pattern matching of rules is best illustrated by an example, Figure 7.2.4c

FIGURE 7.2.4c <u>Example Rule</u>

```
(srule105          (pattern
                   (descriptors  (n is-a = line-terminal)
                     (l is-a = line))
                   (relators     (n has-system = l)
                     (n source-of =a)))
      (action
                          (assert      (l inflow = n))))
```

The pattern is divided into Descriptors and Relators to simplify pattern matching. It is fairly easy, however, to make this transparent to the user. Descriptors describe symbolic attribute values and relators relate one entity to another.

The aim of the pattern matching is to produce a Binding Frame for the variables of the pattern, in this case n and l. The Binding Frame gives the different possible combinations of the variable values which are positive matches to the Facts Database, here the Structural Representation. For example, two Binding Frames which are produced by matching to the Two Phase Splitter are:

```
((n n7)        ((n n1)
 (l L2)         (l L1)
 (a a7))        (a a5))
```

Rule Actions

Once the Binding Frame has been established, the actions of the rule, here assertions, are executed according to the different possible rule-variable value combinations; the rule is said to **fire**. This process adds new assertions to the Facts Database, adding new slots and values to existing objects. If a variable in the assertions is not bound in the Binding Frame then it is assumed to refer to a new object. This new object must have a defined **is-a** slot to tell the system what it is; the object is created and attributed to according to the other action assertions and is added to the Facts Database. This procedure is used for the creation of some variables and the creation of all event objects.

In the above example rule and Binding Frame, the assertions below would be added to the Facts Database:

```
(L2 inflow = n7)    (L1 inflow = n1)
```

7.2.4.3 Rule Processing

In the experimentation a forward chaining strategy was used as the application naturally involves the synthesis of conclusions, namely structure statements, events and causal statements. In the Case Study the amount of matching that had to be done was nevertheless considerable as the search space for even such a simple example is a very large one. For any complex analysis this exhaustive rule processing may not be feasible and a combination of several search strategies may best be employed, in conjunction with heuristic and meta-rules. One possible alternative would be a backward chaining strategy driven by a specified final consequence, eg for the top event of a Fault Tree.

Each of the rule bases to be applied to the facts database is treated in the same fashion: the body of the rule is repeatedly iterated until no more rules fire. The inclusion of NOT clauses in the rule base occasionally calls for the retraction of some assertions. A simple conflict resolution strategy of single pattern firing is used to eliminate looping, eg one rule makes an assertion, another retracts it and then the first makes it again. Any rule is only allowed to fire once for each pattern match.

7.2.5 **Evaluating the Success of the Design**

In the evaluation of an automated method of fault analysis it is necessary to have some measure of the degree of completeness of the results. For a case study, the standard against which the automatic analysis would be set is ultimately the standard of the same analysis carried out by a human expert. One possible measure of the degree of completeness would be a percentage arrived at through calculating:

$$\frac{\text{number of failure modes of system from automated analysis}}{\text{number of failure modes of system from standard analysis}} \quad * \quad 100$$

In each case the number of failure modes of the system could be calculated by elucidating all cut sets of all top events in a fault tree analysis.

The consistently achievable minimum percentage score required of a computer is a matter of judgement. The subject concerns safety and loss prevention and the risks may be considerable, requiring a high degree of confidence that all significant risk potentials are included. However, this is also problematic in studies conducted by people. It is not inconceivable that a computer would achieve a score of greater than 100%, thereby outperforming a human analyst in this particular test. The computer has two main advantages, a perfect memory and tireless high speed concentration.

7.3 GENERALIZATION

7.3.1 **Generalizing to Other Applications.**

As Expert Systems work by a process of inference rather than determination it is difficult, from a single case study and without constructing an operational system, to assess how the system will perform when other cases are analysed. The construction of the architecture was intended to provide a facility for analysing a range of cases, but the knowledge contained within the knowledge bases and hierarchies was more restricted.

The construction of the Knowledge Bases was, however, done with a view to a general picture of process plant, this is particularly true of the key part of the system: the representation of the Structure of Process Plant. The KBS has been extended to cover the whole of the Ammonia Let Down System of Appendix 1, including the Syngas Desorber of Figure 7.3.1a, and a Structural Representation was defined for the Hydrocarbon Reactor - Figure 7.3.1b.

Whilst the description of devices, pipes, control system items etc is relatively straightforward, representation of complex plant units requires an in-depth analysis of the internals of those units. As demonstrated in the ALDS two phase splitter, and in the examples overleaf, it is proposed to use the system of nodes devised by Lihou [1980c etc]. Here, nodes are used to represent locations of concepts of interest, such as inlets, outlets, streams, interfaces, plates, variables, bulk phases etc.

FIGURE 7.3.1a Syngas Desorber - Graphical Representation

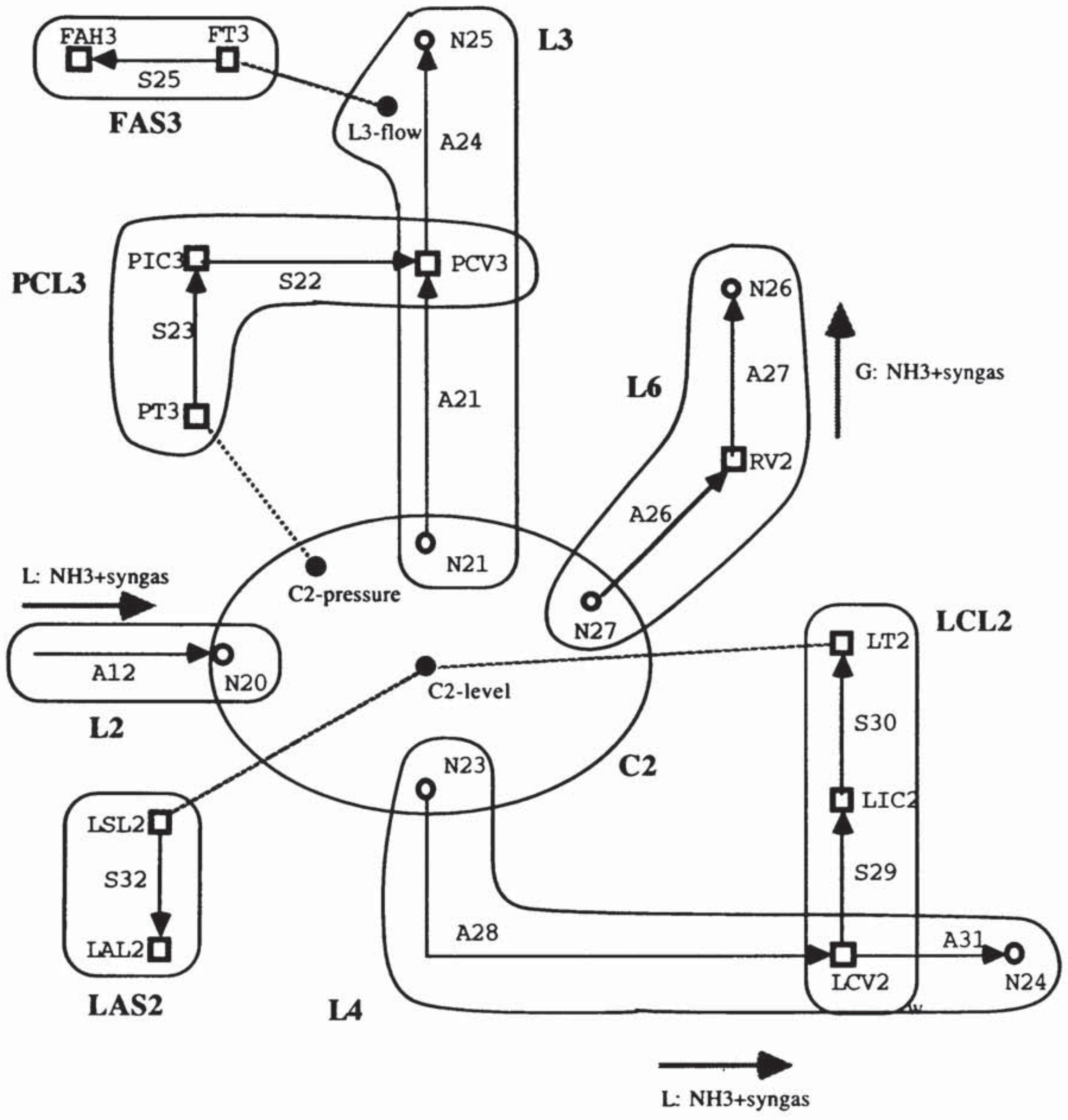
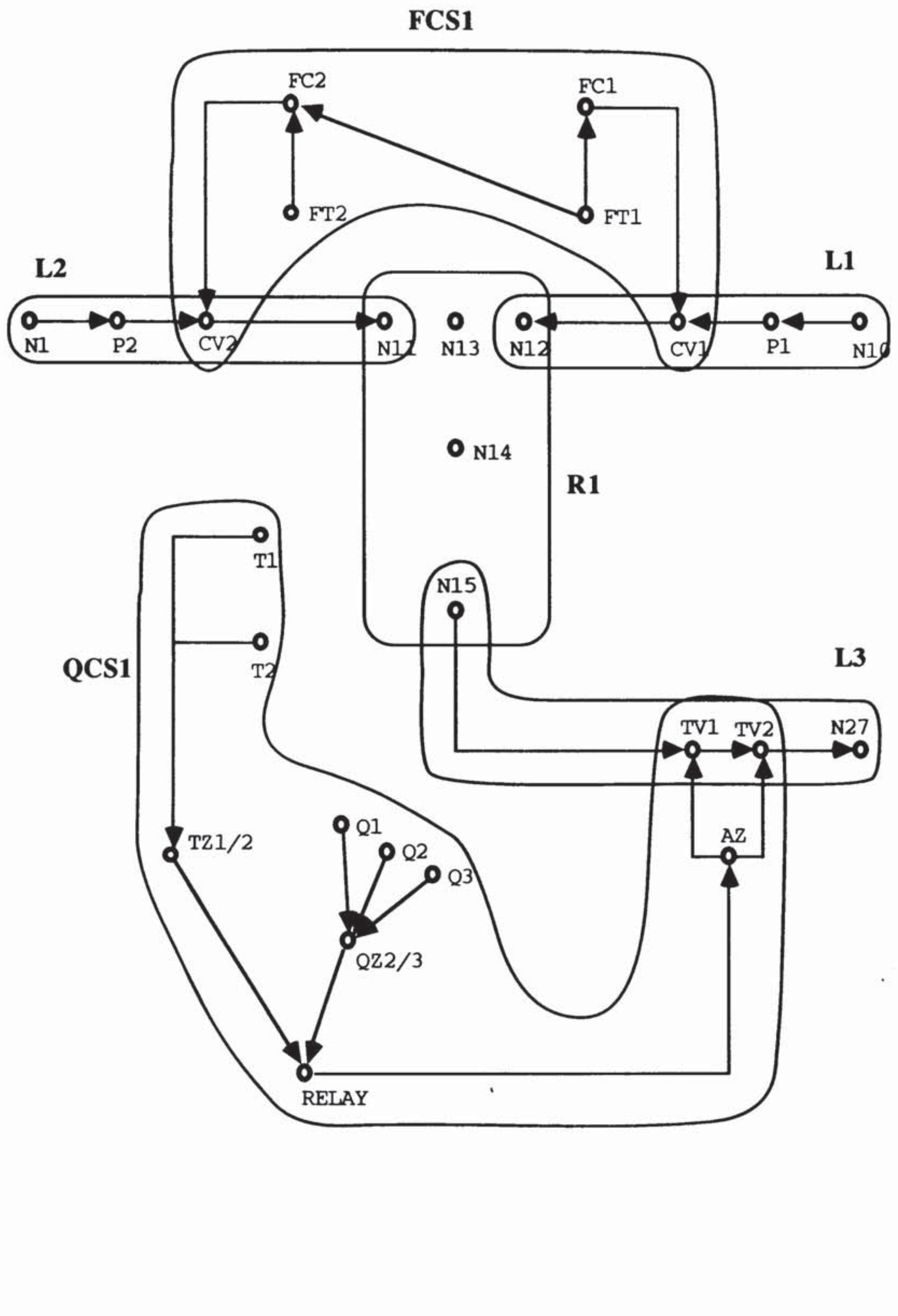


FIGURE 7.3.1b Hydrocarbon Reactor - Graphical Representation



7.3.2 **Constructing an Operational KBS**

7.3.2.1 Tools

In order to construct an operational KBS the appropriate tools are required. For the purposes of experimentation and to provide as much flexibility as possible in the selection and construction of symbol-structures, LISP was chosen for the experimentation described above. At the time of the initiation of the research, purpose built tools were either too expensive or too primitive. The remarkable decline in the cost of computer power in the last five years has led to the availability of powerful workstations supporting integrated knowledge engineering environments. It is obviously much more economical of effort to make use of an appropriate environment than to use a simple high level language, such as LISP, alone.

When constructing the KBS using LISP, the author was free to use whatever representations and procedures which were necessary and feasible. Any tool (which would be written in LISP) will provide more restricted facilities. The KBS environment must satisfy the following requirements.

1) Representation

A Frame based system is required with the facility to represent and manage stereotype objects in a multiple inheritance hierarchy.

2) Rules

For the production of the Causal Representation, a simple forward chaining inference engine is all that is required. There must also be a means of partitioning rule bases, retraction of assertions, provision of special forms in the rules and the creation of new object frames.

3) Embedding

Because of the special forms in the representation and rules it is necessary to write procedures for executing and accessing these forms. This requires the ability to write LISP code. It is necessary to be able to embed LISP procedures in rules and to embed rule processing of independent knowledge bases in LISP procedures which define executive actions.

4) Management

The tool must provide facilities for knowledge management and graphical, interactive development and display.

7.3.2.2 KBS Construction

The construction of an operational KBS would involve the following steps:

- Construct an architecture as described in Chapter Six, with the facility for synthesizing fault trees. Fault trees provide the most comprehensive and well-defined form of causal analysis, thereby facilitating the comparison of case study results with the results of the expert system.

- Select an example case study, eg the ALDS, and construct the necessary knowledge bases for that case study.

- Evaluate the results in an interactive fashion. The case study specifies the events required and their cause-effect relationships.

- Generalise the system by a process of successive application to various representative case studies. Each application involves extension and refinement of the knowledge bases.

7.3.2.3 The Expert in the Expert System

In the proposed architecture, facilities are provided for the expression and use of high-level, experiential or heuristic knowledge that characterises the expert approach to the problem. It is necessary, however, to make use of detailed or low-level knowledge to provide robustness. The expert can fall back on underlying theoretical knowledge of chemical process operations when a difficult or novel situation is encountered. When they are available, it may be desirable to make use of mathematical models or numerical simulations.

A human expert is able to justify his solution to a given problem and a truly Expert system should be able to do the same. This is commonly done by reiteration of the rules used to arrive at the goals and the facts upon which those rules operated. Here, justification of a given pattern of cause-effect would involve breaking down that pattern into the detail of the parts of the causal network from which it was formed. This would be supported by the causal rules used in generating those parts of the network and the original plant representations to which they were applied.

CHAPTER EIGHT

CONCLUSION

Conventional data processing or algorithmic approaches to the problem at hand seem to be doomed to failure when it comes to real-world applications. This is because they are, generally speaking, deterministic in nature and are therefore baulked by their own complexity in attempting to produce exact models of system behaviour. This may also be true of the attempts in the AI community to model the behaviour of physical systems using "naive" or Qualitative Physics. These methods approach the complexity of mathematical modelling.

Rather than produce a fault analysis deterministically, the expert system approach involves the implication of factors in the end analysis. Once the architecture of the KBS, which is capable of solving the problem with the necessary knowledge, has been decided upon, the inclusion of knowledge is often an ad hoc process. Chunks of knowledge are expressed, perhaps in the form of rules, and the development of the knowledge bases proceeds by an iterative process; analysing the performance of the system and interactively developing the knowledge bases. In this instance case studies exist - those providing fault trees being most appropriate - on which to base and evaluate knowledge base construction.

This research has shown that the problem at hand is a very complex one, involving, in principle, the whole domain of chemical engineering. As such it requires a complex solution. The solution presented here involves the division of the fault analysis problem into four stages:

- Representation of the plant under consideration
- Synthesis of events of interest that may occur
- Synthesis of cause-effect that may occur
- Structured production of the fault analysis

High level rules and procedures for conducting fault analyses exist. One example is given in the rules for HAZOP described in Section 2.4. Another is the set of fault tree synthesis guidelines described in Section 2.5. These rules are generally too abstract to be directly applicable to the production of a fault analysis from a structural description of the system. They require behavioural models of the system under analysis - this is provided by the causal representation and valid event set of the proposed design.

This research has led to the development of an architecture for an expert system. Future work would involve the following steps.

- Selection of a suitable KBS building environment.
- Implementation of the proposed architecture.
- Interactive development of the knowledge bases using case studies.

It is necessary to evaluate the success of the expert system in solving fault analysis problems. The benchmark for this test is ultimately the performance standard of human analysts who are recognised as experts. Unfortunately, mistakes in general, and omissions in particular, are traits of all humans in problem solving. It is not inconcievable, therefore, that an expert system could actually outperform a human expert according to some measures of success; the computer has two main advantages, a perfect memory and tireless high speed concentration.

REFERENCES

Addis TR (1985) Designing Knowledge Based Systems
Kogan-Page, London

Aggarwal KK &Rai S (1978) 'Symbolic Reliability Evaluation Using Logical Signal Relations'
IEEE Transactions on Reliability, Vol R-27, p 202

Aggarwal KK (1979) 'Comments on "An Efficient Simple Algorithm for Fault Tree Automatic Synthesis from the Reliability Graph"'
IEEE Transactions on Reliability, Vol R-28, No 4

Aggarwal KK, Chopra YC &Bajwa JS (1982) 'Reliability Evaluation by Network Decomposition'
IEEE Transactions on Reliability, Vol R-31, No 4, p 355

Aikins JS (1980) 'Representation of Control Knowledge in Expert Systems'
Proceedings 1st NCAI, Stanford, August, p121

Allan RN, Rondiris IL & Fryer DM (1981) 'An Efficient Computational Technique for Evaluating the Cut/Tie Sets and Common Cause Failures of Complex Systems'
IEEE Transactions on Reliability, Vol R-30, No 2

Allen DJ & Rao MS (1980) 'New Algorithms for the Synthesis and Analysis of Fault Trees'
Ind. Eng. Chem. Fund., Vol 19, p79

Allen JF (1983) 'Maintaining Knowledge About Temporal Intervals'
Commun. Assoc. Computing Machine, Vol 26, p 832

Allen JF (1984) 'General Theory of Action and Time'
Artificial Intelligence, Vol 23, No 2

Alty JL & Coombs MJ (1984) Expert Systems, Concepts and Examples
NCC Publications, Manchester

Andow PK & Lees FP (1974) 'Process Plant Alarm Systems: General Considerations'
Loss Prevention and Safety Promotion in the Process Industries. 1st Annual Symposium (Elsevier, Amsterdam)

Andow PK & Lees FP (1975) 'Process Computer Alarm Analysis: Outline of a method based on List Processing'
Transactions IChemE, Vol 53, October 1975, p 195

Andow PK & Lees FP (1978) 'Real Time Analysis of Process Plant Alarms' in [Apostolakis 1978]

Andow PK (1980a) 'Difficulties in Fault Tree Synthesis for Process Plant' IEEE Transactions on Reliability, Vol R-29, No 1

Andow PK,Lees FP & Murphy CP (1980b) 'The Propagation of Faults in Process Plants: A State of the Art Review' Chemical Process Hazards With Special Reference to Plant Design IChemE Symposium Series No 58, p 225 (IChemE, Rugby)

Andow PK (1985a) 'Fault Diagnosis using Intelligent Knowledge Based Systems', Chemical Engineering Research & Design, Vol 63, November 1985

Andow PK (1985b) 'Fault Diagnosis using Intelligent Knowledge Based Systems' PSE '85: The Use of Computers in Chemical Engineering, IChemE Symposium Series No 92 (Pergamon, Oxford)

Andow PK (1985c) 'Improvement of Operator Reliability using Expert Systems', Reliability '85

Anon. (1977) 'References on Failure Data' in Proceedings 2nd International Symposium on Loss Prevention and Safety Promotion in the Process Industries , Heidelberg, (DECHEMA, Frankfurt)

Ankakora SN, Engel GFM & Lees FP (1971) 'Some Data on the Reliability of Instruments in the Chemical Plant Environment' The Chemical Engineer, Nov 1971, p 396

Apostolakis G, Garriba S & Volta G (1978) Synthesis and Analysis Methods for Safety and Reliability Studies (Plenum Press, London)

Barbuceanu M (1984) 'Object Centered Representation and Reasoning: An Application to Computer Aided Design' SIGART Newsletter, Vol 87, p 33

Barlow RE, Fussell JB & Singapurwalla (1975a) Reliability and Fault Tree Analysis (SIAM, Philadelphia)

Barlow RE & Lambert HE (1975b) 'Introduction to Fault Tree Analysis' in [Barlow 1975a]

Barlow RE & Proschan F (1975c) Statistical Theory of Reliability and Life Testing Probability Models
(Holt, Rinehart and Winston, New York)

Barr A & Feigenbaum E (eds) (1981) The Handbook of Artificial Intelligence, Volumes 1,2 & 3
William Kaufman Inc, Los Altos, California

Basden A & Kelly BA (1981) 'DART: An expert system for Computer Fault Analysis'
Proceedings of the 7th International Joint Conference on Artificial Intelligence, p 843

Baybutt P (1986) 'Decision Support Systems for Risk and Safety Analysis'
Hazards IX, Proceedings of the Symposium on Process Industry Hazards, UMIST, April 1986

Beazley B (1986) 'A Guide to AI Tools for Process Control'
Expert Systems User, July, p 8

Becker JV & Hill R (1979a) 'Fundamentals of Interlock Systems'
Chemical Engineering, Oct 8, 1979, p 101

Becker JV (1979b) 'Designing Safe Interlock Systems'
Chemical Engineering, Oct 15, 1979, p 103

Begg V (1984) Developing Expert CAD Systems
Kogan-Page, London

Bellingham B & Lees FP (1977) 'The Detection of Malfunction Using a Process Control Computer'
Trans IChemE, Vol 55, No 1

Bendell (1978) 'An Overview of Collection, Analysis and Application of Reliability Data in the Process Industries'
IEEE Transactions on Reliability, Vol R-37, No 2, p 132

Bengiamin NN, Bowen BA & Schenk KF (1976) 'An Efficient Algorithm for Reducing the Complexity of Computation in Fault Tree Analysis'
IEEE Transactions on Nuclear Science, Vol NS-23, No 5

Bennetts RG (1975) 'On the Analysis of Fault Trees'
IEEE Transactions on Reliability, Vol R-24, Aug, p 175

Bennetts RG	(1976)	A Comment on the Reliability Evaluation of Software in: Henley EJ & Lynn JW (eds) NATO Advanced Study Institute on Generic Techniques in System Reliability Assessment Noordhoff, Amsterdam
Berenblut BJ & Whitehouse HB	(1977)	'A Method for Monitoring Process Plant Based on a Decision Table Analysis' The Chemical Engineer, March 1977
Bignell V & Fortune J	(1984)	Understanding Systems Failures (Manchester University Press)
Bo K	(1982)	'Human Computer Interaction' IEEE Computer, November 1982
Bobrow D & Winograd T	(1977)	'KRL: Knowledge Representation Language' Cognitive Science, Vol 1, No 1
Bobrow DG	(1984)	'Qualitative Reasoning About Physical Systems: An Introduction' Artificial Intelligence, Vol 24, p1
Boley H	(1983)	'Artificial Intelligence Languages and Machines' Technology and Science of Informatics, Vol 2, No 3
Booch G	(1986)	'Object Oriented Development' IEEE Transactions on Software Engineering, Vol SE-12, No 2
Boose J	(1984)	'Personal Construct Theory and the Transfer of Human Expertise' Proceedings National Conference on Artificial Intelligence Austin, Texas
Booth SH	(1986)	PC-CAFOS User Manual Lihou Loss Prevention Services Ltd, Birmingham
Booth SH & Jones MC	(1987)	'Rule-Based Programming for Operability Studies' IChemE Annual Research Meeting, Nottingham 9-4-87
Brachman RJ	(1979)	'On the Epistemological Status of Semantic Networks' in: Findler NV (ed), Associative Networks: Representation and Use of Knowledge by Computers, Academic Press, New York

Brachman RJ	(1983)	'What is-a and isn't: An Analysis of Taxonomic Links in Semantic Networks' IEEE Transactions on Computers, Vol C-32, p 30
Bremer	(1988)	"Expert Systems in Business: a British Perspective." J. Expert Systems, May 1988, p104-117.
Brodsky S & Tyle N	(1984)	'Knowledge Based Expert Systems for Power Engineering' Proceedings 15th Modelling and Simulation Conference,Pittsburgh
Browning RL	(1970)	'Finding the Critical Path to Loss' Chemical Engineering, Jan 26 1970
Browning RL	(1972)	'Human Factors in the Fault Tree' Chemical Engineering Progress, June, p72
Bruce BC	(1972)	'A Model for Temporal References and its Application in a Question Answering Program' Artificial Intelligence, Vol 3, p 1
Buchanan b & Shortliffe E (ed)	(1984a)	Rule Based Expert Systems Addison-Wesley, Reading, Massachusetts
Buchanan B	(1984b)	'Partial Bibliography of Work on Expert Systems' SIGART Newletter, No 84, April
Camarda P & Corsi F	(1978)	'An Efficient Simple Algorithm for Fault Tree Automatic Synthesis from the Reliability Graph' IEEE Transactions on Reliability, Vol R-27, No 3
Chandrasekaran B & Mittal S	(1982)	'Deep Versus Compiled Knowledge Approaches to Diagnostic Problem Solving' AAAI-82, p 349
Charniak E	(1978)	'On the Use of Framed Knowledge in Language Comprehension' Artificial Intelligence, Vol 11, No 3
Charniak E, Riesbeck C & McDermott D	(1980)	Artificial Intelligence Programming Lawrence Earlbaum Associates, Hillsdale, New Jersey

Chester D, Lamb D &Dhurjati P (1984) 'Rule Based computer alarm analysis in Chemical Process Plants'
Proceedings of the 7th annual conference on computer technology, MICRO-DELCON 84, IEEE, March 1984, p 22

Chinneck JW (1985) 'On Systems Theory and Models of Heat Flow'
IEEE Transactions on Systems, Man and Cybernetics
Vol SMC-15, No 3, p 423

CIA (1977) Chemical Industries Safety and Health Council
A Guide to Hazard and Operability Studies
(Chemical Industries Association, London)

Clancey WJ & Letsinger R (1981) 'NEOMYCIN: Reconfiguring a Rule Based Expert System for Application to Teaching'
Proceedings International Joint Conference on Artificial Intelligence IJCAI-81, p 829

Clancey WJ (1983) 'The Epistemology of a Rule Based Expert System - a Framework for Explanation'
Artificial Intelligence, Vol 20, p 215

Clancey WJ (1985) 'Heuristic Classification'
Artificial Intelligence, Vol 27, p 289

Clark KL & McCabe FG (1982) 'PROLOG: A Language for Implementing Expert Systems'
In: Hayes JE (ed), Machine Intelligence 10
Harwood, Chichester

Cohen BL (1977) 'Structural Pattern Recognition'
Artificial Intelligence, Vol 9, No 3, p 223

Cohen H (1984) 'Space Reliability Technology - An Historical Perspective'
IEEE Transactions on Reliability, Vol R-33, No 1, p 36

Coombs MJ (ed) (1984) Developments in Expert Systems
Academic Press

Coppola A (1984) 'Reliability Engineering of Electronic Equipment - An Historical Perspective' IEEE Transactions on Reliability, Vol R-33, No 1, p 29

Cox BJ (1986) Object Oriented Programming - An Evolutionary Approach
Addison-Wesley, Reading, Massachusetts

Cummings GE (1975) 'Application of the Fault Tree Technique to a Nuclear Reactor Containment System'
in [Barlow 1975a]

Cummings DL, Lapp SA & Powers GJ (1983) 'Fault Tree Synthesis from a Directed Graph Model for a Power Distribution Network'
IEEE Transactions on Reliability, Vol R-32, No 2

Curry RE & Ephrath AR (1976) 'Monitoring and Control of Unreliable Systems'
in Sheridan TB (ed), Monitoring Behaviour and Supervisory Control Plenum Press, New York

D'Ambrosio B (1985) 'Software Review - Golden Common Lisp'
BYTE, December, p 317

Davis R (1977) An Overview of Production Systems
in: Programming Tools for Knowledge Representation

Davis R (1979) 'Interactive Transfer of Expertise: Acquisition of New Inference Rules'
Artificial Intelligence, Vol 12, p 121

Davis R & Lenat D (1980a) Knowledge Based Systems in Artificial Intelligence
McGraw-Hill, New York

Davis R (1980b) 'Metarules: Reasoning About Control'
Artificial Intelligence, Vol 15, p 179

Davis R (1982a) 'Diagnosis Based on Description of Structure and Function'
AAAI-82, p 137

Davis R (1982b) 'Expert Systems: Where are we and where do we go from here?'
AI Magazine, Vol 3, No 2, p3

Davis R (1983) 'Diagnosis via Causal Reasoning: Paths of Interaction and the Locality Principle'
International Journal of Man-Machine Studies, Vol 15

Davis R (1984) 'Diagnostic Reasoning Based on Structure and Behaviour'
Artificial Intelligence, Vol 24, p 347

Dhillon BS (1988) 'Chemical System Reliability - A Review'
IEEE Transactions on Reliability, Vol R-37, No 2, p 199

DOW Chemical Company (1981) Fire and Explosion Index, 5th Edition
CEP Technical Manual, AIChE, New York

Dunglinson C & Lambert H (1983) Interval Reliability for Initiating and Enabling Events'
IEEE Transactions on Reliability, Vol R-32, No 2

Dym CL (1984) 'Expert Systems: New Approaches to Computer-Aided Engineering'
Proceedings 25th AIAA-ASME-ASCE-AHS Structures, Structural Dynamics and Materials Conference, California

Edwards E & Lees FP (1972) Man and Computer in Process Control
IChemE, London

Embrey DE (1986a) 'Improving Human Reliability in Abnormal Conditions by the Application of a Decision Support System'
Personal Communication

Embrey DE (1986b) 'Support for Decision Making and Problem Solving in Abnormal conditions in Nuclear Power Plant'
Personal Communication

Esary JD & Ziehms H (1975) 'Reliability Analysis of Phased Missions'
in [Barlow 1975a] p 213

Evans RA (1978) 'Automatic Fault Tree Generation: Why, What and How'
IEEE Transactions on Reliability, Vol R-27

Fain J (1981) The Rosie Language Reference Manual
Technical Report N-1647-ARPA
Rand Corporation, Santa Monica, California

Fichtelman M (1985) 'The Expert Mechanic'
BYTE, June, p 205

Findler N & Chen D (1973) 'On the Problem of Time, Retrieval of Temporal Relations, Causality and Coexistence'
International Journal of Computing and Information Science, Vol 2, No 3

Finetti B (1974) Theory of Probability
(John Wiley & Sons, New York)

Fink PK, Lusth JC & Duran JW (1985) 'A General Expert System Design for Diagnostic Problem Solving'
IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-7, No 5

Fothergill CDH (1975) The Collection, Storage and Use of Equipment Performance Data for the Safety and Reliability
Assessment of Nuclear Power Plants
(International Atomic Energy Agency, Vienna, Austria)

Forbus KD (1984) 'Qualitative Process Theory'
Artificial Intelligence, Vol 24, p85

Forgy CL (1981) OPS5 Users Manual
Technical Report CMU-CS-81-135
Carnegie-Mellon University, Pittsburgh, Philadelphia

Fox MF (1979) 'On Inheritance in Knowledge Representation'
Proceedings 6th International Joint Conference on Artificial Intelligence, Tokyo

Frost RA (1986) Introduction to Knowledge Base Systems
Collins, London

Fussell JB (1973a) 'A Formal Methodology for Fault Tree Construction'
Nuclear Science and Engineering, Vol 52, p 421

Fussell JB (1973b) Synthetic Tree Model, a Formal Methodology for Fault Tree Construction, Report ANCR-1098
(Aerojet Nuclear Co., Idaho Falls, Idaho, USA)

Fussell JB, Powers GJ & Bennetts RG (1974) 'Fault Trees - A State of the Art Discussion'
IEEE Transactions on Reliability, Vol R-23, No 1

Fussell JB (1975) 'Computer-Aided Fault Tree Construction for Electrical Systems', in [Barlow 1975a]

Fussell JB (1976) 'Fault Tree Analysis, Concepts and Techniques'
[Henley 1976]

Fussell JB & Burdick GR (eds) (1977) Nuclear System Reliability Engineering and Risk Assessment
(SIAM, Philadelphia)

Fussell JB	(1984)	'Nuclear Power System Reliability: A Historical Perspective' IEEE Transactions on Reliability, Vol R-33, No 1, p 41
Garriba S et al	(1977)	'Efficient Construction of Minimal Cut Sets from Fault Trees' IEEE Transactions on Reliability, Vol R-26, No 2
Genesereth MR	(1982)	'Diagnosis using Hierarchical Design Models' Proceedings AAAI -82, p 278
Genesereth MR	(1983)	'The DART Project' Artificial Intelligence magazine, Fall 1983, p 85
Genesereth MR	(1984)	'The Use of Design Descriptions in Automated Diagnosis' Artificial Intelligence, Vol 24, p 411
Gentner D & Stevens AL	(1983)	Mental Models Lawrence Earlbaum, Hillsdale, New Jersey
Gibson SB	(1976)	'The Design of Chemical Plants Using Hazard Analysis' Proceedings of the Symposium on Process Industry Hazards IChemE Symposium Series No 47 (IChemE, Rugby)
Gibson SB	(1979)	'The Quantitative Measurement of Process Safety' in Chemical Process Hazards With Special Reference to Plant Design VI\ IChemE Symposium Series No 49 (IChemE, Rugby), p 1
Gigch JP	(1986)	'Modelling, Meta-modelling and Taxonomy of System Failures' IEEE Transactions on Reliability, Vol R-35, No 2, p 131
Gold Hill Computers Inc	(1984)	Golden Common Lisp Reference Manual Gold Hill Computers
Goldberg A & Robson D	(1984)	Smalltalk-80; The Language and its Implementation Addison-Wesley, Reading, Massachusetts
Gomez F	(1979)	'Knowledge Organization and Distribution for Diagnosis' IEEE Transactions on Systems, Man and Cybernetics, January
Gomez F, Chandrasekaran B & Smith J	(1981)	'Knowledge Organization and Distribution for Medical Diagnosis' IEEE Transactions on Systems, Man and Cybernetics, Vol SMC-11, No 1, p 34

Green AE & Bourne AJ (1972) Reliability Technology
(Wiley-Interscience, London)

Green AE (1976) 'A Review of System Reliability Assessment'
[Henley 1976], p 183

Green AE (ed) (1982) High Risk Safety Technology
(John Wiley & Sons, New York)

Gupta Y (1982) 'Software Safety Systems: A Critical Appraisal'
Proceedings 7th Advances in Reliability Technology Symposium NCSR (Chamelion Press, London)

Haasl DF (1965) Advanced Concepts on Fault Tree Analysis
System Safety Symposium, June 8-9, 1965 (The BOENG Company, Seattle, Washington)

Haasl DF & Vesely WE (1981) Fault Tree Handbook
NUREG-0492 (Office of Nuclear Regulatory Research, US Nuclear Regulatory Commission, Washington DC)

Hamner W (1972) Handbook of System and Product Safety
(Prentice-Hall, Englewood Cliffs, New Jersey)

Hart P (1982) 'Direction for AI in the Eighties'
Sigart Newsletter, No 79

Harteley RT (1981) 'How Expert Should an Expert System Be?'
Proceedings International Joint Conference on Artificial Intelligence IJCAI-81, p 862

Hartely RT (1984) 'CRIB: Computer Fault Finding through Knowledge Engineering'
Computer, Vol 17, No 3, March 1984

Hayes PJ (1977) 'In Defence of Logic'
Proceedings 5th Joint Conference on Artificial Intelligence

Hayes-Roth B (1985) 'A Blackboard Architecture for Control'
Artificial Intelligence, Vol 26, p 251

Hayes-Roth F, Waterman DA & Lenat DB (1983) Building Expert Systems
Addison-Wesley, Reading, Massachusetts

Henderson P	(1980)	Functional Programming Prentice-Hall, Englewood Cliffs, New Jersey
Hendrix GG	(1975)	'Expanding the Utility of Semantic Networks Through Partitioning' 4th International Joint Conference on Artificial Intelligence, Tblisi, USSR
Hendrix GG	(1978)	'Developing a Natural Language Interface to Complex Data' ACM Transactions on Database Systems, Vol 3, No 2
Henley EJ & Lynn JW (eds)	(1976)	NATO Advanced Study Institute on Generic Techniques in System Reliability Assessment (Noordhoff, Amsterdam)
Henley EJ & Kumamoto H	(1977)	'Comment on: Computer-Aided Synthesis of Fault Trees' IEEE Transactions on Reliability, Vol R-26, p 316
Henley EJ & Kumamoto H	(1981)	Reliability Engineering And Risk Assessment (Prentice-Hall, Englewood Cliffs, New Jersey)
Henley EJ & Kumamoto H	(1985)	'Designing for Reliability and Safety Control' in PSE '85: The Use of Computers in Chemical Engineering, IChemE Symposium Series No 92, p 651.(Pergamon, Oxford)
Hensley G	(1985)	'The Application of PRA In Demonstrating the Adequacy of Safety in the Design of Nuclear Chemical Plants' ANS/ENS Topical Meeting: Probabilistic Safety Methods and Applications, San Fransisco, USA
Heslinga G	(1983)	'Human Reliability Analysis Using Event Trees' Kema Scientific and Technical Reports, Vol 1, No 3, p 19
Hewitt C	(1977)	'Viewing Control Structures as Patterns of Passing Messages' Artificial Intelligence, Vol 8, No 3
Himmelblau DM & Bischoff KB	(1968)	Process Analysis and Simulation: Deterministic Systems (John Wiley & Sons, New York)
Himmelblau DM	(1978)	Fault Detection and Diagnosis in Chemical and Petrochemical Processes (Elsevier, Amsterdam)
Hix AH	(1977)	'Safety and Instrumentation Systems' Loss Prevention , Vol 11, p4

Hollo E & Taylor JR	(1977)	Algorithm and Programs for Consequence Diagram and Fault Tree Construction RISO-M-1907 (Danish Atomic Energy Commission, Roskilde, Denmark)
Hughes J	(1986)	Why Functional Programming Matters Programming Methodology Group Memo PMG-40
Humphreys P	(1986)	'Intelligence in Decision Support' New Directions in Research and Decision Making (North-Holland, Amsterdam)
Hwang CL, Tillman FA & Lee MH	(1981)	'System Reliability Evaluation Techniques for Complex / Large Systems, a Review' IEEE Transactions on Reliability, Vol R-30, No 5, p 416
IEE	(1983)	The Why and How of Expert Systems - A Review for Engineers IEEE Digest No 1983/21a
ISGRA	(1985)	International Study Group on Risk Analysis Risk Analysis in the Process Industries IChemE, Rugby
Iwasaki Y & Simon HA	(1986a)	'Causality in Device Behaviour' Artificial Intelligence, Vol 29, p3
Iwasaki Y & Simon HA	(1986b)	'Theories of Causal Ordering: Reply to de Kleer and Brown' Artificial Intelligence, Vol 29, p 63
Joller JM	(1982a)	'Maintenance of Reliable Non-coherent Systems' Proceedings 7th Advances in Reliability Technology Symposium NCSR (Chamelion Press, London)
Joller JM	(1982b)	'Constructing Fault Trees by Stepwise Refinement' IEEE Transactions on Reliability, Vol R-31, No 4
Jones MC & Lihou DA	(1987)	'CAFOS - The Computer Aid for Operability Studies' IChemE Symposium Series, No 97, p 249
Kahn G, Nowlan S & McDermott J	(1985)	'Strategies for Knowledge Acquisition' IEEE Transactions on Pattern Analysis and Machine Intelligence Vol PAMI-7, No 5

Kahn KM & Gory AG	(1977)	'Mechanizing Temporal Knowledge' Artificial Intelligence, Vol 9, No 2, p 87
Kane LA	(1986)	'AI and MAP in the Process Industries' Hydrocarbon Processing, June, p 55
Kaplan RM	(1972)	'Augmented Transition Networks as Psychological Methods of Sentence Comprehension' Artificial Intelligence, Vol 3, No 4
Kaplan SJ	(1984)	'The Industrialization of Artificial Intelligence: From By-line to Bottom Line' The AI Magazine, Vol 5, No 2
Kinchin GH	(1978)	'Plant Reliability and Safety' Major Chemical Hazards Symposium (UKAEA, Harwell)
King CF & Rudd DF	(1972)	'Design and Maintenance of Economically Failure\Tolerant Processes' AIChE Journal, Vol 18, p 257
Klaassen PL	(1979)	'The Importance of Software/Hardware for Safe Processing' Loss Prevention, Vol 12, p 118
Klahr P & Waterman DA (eds)	(1986)	Expert Systems: Techniques, Tools and Applications Addison-Wesley, New York
Kleer J & Brown JS	(1984)	'A Qualitative Physics Based on Confluences' Artificial Intelligence, Vol 24, p 7
Kleer J & Brown JS	(1986)	'Theories of Causal Ordering' Artificial Intelligence, Vol 29, p 33
Kletz TA	(1978)	'Practical Applications of Hazard Analysis' Chemical Engineering Progress, Vol 74, No 10, p 47
Kletz TA	(1981)	'Plant Instruments: Which Ones Don't Work and Why' Loss Prevention, CEP, AIChE, Vol 14, p 108
Kletz TA	(1985)	HAZOP and HAZAN: Notes on the Identification and Assessment of Hazards (IChemE, Rugby)

Knuth DE	(1981)	<u>The Art of Computer Programming</u>, Vols 1,2 & 3 Addison-Wesley, Reading, Massachusetts
Konopasek M & Jayaraman J	(1984)	<u>Expert Systems for Personal Computers - The TK!Solver Approach</u> BYTE, May, p 137
Korn feld WA	(1979)	'Pattern Directed Invocation Languages' <u>BYTE</u>, Vol 4, No 8
Kuipers B	(1984)	'Commonsense Reasoning About Causality: Deriving Behaviour from Structure' <u>Artificial Intelligence</u>, Vol 24, p 169
Kumamoto H & Henley EJ	(1979)	'Safety and Reliability Synthesis of Systems With Control Loops', <u>AIChE Journal</u>, Vol 25, No 1, p 108
Kumamoto H, Henley EJ & Inoue K	(1981)	'Signal Flow Based Graphs for Failure Mode Analysis of Systems With Control Loops' <u>IEEE Transactions on Reliability</u>, Vol R-30, No 2
Kumamoto H et al	(1984)	'Applications of Expert System Techniques to Fault Diagnosis' <u>The Chemical Engineering Journal</u>, Vol 29, p 1
Lambert HE	(1979)	'Comments on the Lapp-Powers "Computer-Aided Synthesis of Fault Trees"' <u>IEEE Transactions on Reliability</u>, Vol R-28, No 1
Lapp SA	(1976)	<u>Computer-Aided Fault Tree Synthesis</u> MSc Thesis (Carnegie-Mellon University, Pittsburgh)
Lapp SA & Powers GJ	(1977a)	'The Synthesis of Fault Trees' in <u>[Fussell 1977]</u>
Lapp SA & Powers GJ	(1977b)	'Computer-Aided Synthesis of Fault Trees' <u>IEEE Transactions on Reliability</u>, Vol R-26, No 1, p 2
Lapp S & Powers GJ	(1977c)	'Computer-Assisted Generation and Analysis of Fault Trees' in <u>Proceedings 2nd International Symposium on Loss Prevention and Safety Promotion in the Process Industries , Heidelberg</u>, (DECHEMA, Frankfurt)
Lapp SA & Powers GJ	(1979)	'Update of Lapp-Powers Fault Tree Synthesis Algorithm' <u>IEEE Transactions on Reliability</u>, Vol R-28, No 1

Laviron A, Carnino A & Manaranche JC	(1982)	'ESCAF - A New and Cheap System for Complex Reliability Analysis and Computation' IEEE Transactions on Reliability, Vol R-31. No 4
Lawley HG	(1973)	'Hazard and Operability Studies' Chemical Engineering Progress, Vol 8, No 5, p 105
Lawley HG	(1974)	'Operability Studies and Hazard Analysis' Loss Prevention, Vol 8, p 105
Lawley HG	(1976)	'Size Up Plant Hazards This Way' Hydrocarbon Processing, Vol 55, No 4, p 247
Lawley HG	(1980)	'Safety Technology in the Chemical Industry: A Problem In Hazard Analysis With Solution' Reliability Engineering, Vol 1, p 89
Lee WS et al	(1985)	'Fault Tree Analysis, Methods and Applications - A Review' IEEE Transactions on Reliability, Vol R-34, No 3
Lees FP	(1976)	'A Review of Instrument Failure Data' Proceedings: Symposium on Process Industry Hazards IChemE Symposium Series n 47, p73 (IChemE, Rugby)
Lees FP	(1979)	'The Contribution of the Control System to Loss Prevention' Chemical Process Hazards With Special Reference to Plant Design VI, IChemE Symposium Series No 49 (IChemE, Rugby)
Lees FP	(1980a)	Loss Prevention in the Process Industries, Vols 1 & 2 (Butterworth, London)
Lees FP, Andow PK & Murphy CP	(1980b)	'Fault Tree Analysis' Reliability Engineering, Vol 1, p 149
Leveson NG	(1984)	'Software Safety in Computer Controlled Systems' Computer, IEEE, Feb 1984, p48
Levesque H & Mylopoulos J	(1977)	'An Overview of a Procedural Approach to Semantic Networks' Proceedings 5th Joint Conference on Artificial Intelligence

Lewis CM & Hammer JM	(1986)	'Significance Testing of Rules in Rule Based Models of Human Problem Solving' IEEE Transactions on Systems, Man and Cybernetics, Vol SMC-16, No 1, p 154
Lihou DA	(1977)	'Hazard Analysis in the Design of Chemical Plants' British Association for the Advancement of Science 139th Meeting, Section G, Aston University, August 31 1977
Lihou DA & Trepa de Faria FA	(1978)	'Operability Studies and Fault Finding' CHEMPOR '78, Braga, Portugal, 11-15 Sept 1978
Lihou DA	(1979)	'Instrumentation scheme design to aid Fault Finding' Paper H2, Design '79, Aston University, 13-14 Sept 1979, (IChemE Midlands Branch)
Lihou DA	(1980a)	'Inspection and Maintenance Strategies' Paper 11, Seminar on Safety Promotion and Loss Prevention in the Process Industries, Jun 4 1980, (OYEZ-IBC, London)
Lihou DA	(1980b)	'Efficient Use of Operability Studies' Paper 5, Seminar on Safety Promotion and Loss Prevention in the Process Industries, 4/5 June 1980 (Oyez-IBC, London)
Lihou DA	(1980c)	'Computer-Aided Operability Studies for Loss Control' 3rd International Symposium on Loss Prevention and Safety Promotion in the Process Industries, Basle, Switzerland, 15-19 Sept, 1980
Lihou DA	(1980d)	'Fault Trees From Operability Studies' Paper 6, Seminar on Safety Promotion and Loss Prevention in the Process Industries, London, 4/5 Jun 1980 (Oyez-IBC, London)
Lihou DA	(1980e)	'Aiding Process Plant Operators in Fault-finding and Corrective Action' NATO Symposium on Human Detection and Diagnosis of System Failures, Roskilde, Denmark, 4-8 August, 1980 (Plenum Press)
Lihou DA	(1982a)	Design Reliability and Performance of Protective Systems Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1982b)	Ranking of Process Hazards Internal Publication, Chemical Eng Dept, Aston University

Lihou DA	(1982c)	<u>Hazards in Perspective</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1982d)	'Computer-Aided Operability Study' <u>IChemE Loss Prevention Bulletin</u> 051, p 19
Lihou DA	(1982e)	<u>Computer-Aided Operability Studies</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1983a)	<u>Design and Specification for Safety</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1983b)	<u>System Failures</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1983c)	<u>Hazard Assessment</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1983d)	<u>Fault Finding and Corrective Action</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1983e)	<u>Fault Tree Analysis</u> Internal Publication, Chemical Eng Dept, Aston University
Lihou DA	(1983f)	<u>Operability Studies</u> Internal Publication, Chemical Eng Dept, Aston University
Lindsay R et al	(1980)	<u>Applications of AI to Organic Chemistry: The DENDRAL Project</u> McGraw-Hill, New York
Locks MO	(1979)	'Synthesis of Fault Trees: An Example of Noncoherence' <u>IEEE Transactions on Reliability</u>, Vol R-28, No 1
Locks MO	(1980)	'Comments on L&P' <u>IEEE Transactions on Reliability</u>, Vol R-29, p 130
Lu MD	(1985)	An Expert System for Computer Aided Flowsheet Synthesis iin: <u>PSE 85; The Use of Computers in Chemical Engineering</u> IChemE Symposium Series 92, Pergamon, Oxford
Luyben WL	(1982)	<u>Process Modelling, Simulation and Control for Chemical Engineers</u>, International Student Edition (McGraw-Hill, New York)

Maclean C & Wilde P	(1983)	'Knowledge Based Electronic Circuit Diagnosis' Proceedings Expert Systems '83, Technical Conference of the BCS SGES, Cambridge,UK, Dec 1983
Maguire M	(1984)	'An Evaluation of Published Recommendations on the Design of Human Computer Dialogues' International Journal of Man-Machine Studies, Vol 16, p 237
Maigret N et al	(1982)	'Use of Simulation Methods in the Evaluation of Reliability and Availability of Complex Systems' in Proceedings 7th Advances in Reliability Technology Symposium NCSR (Chamelion Press, London)
Mamdani EH	(1982)	'Rule Based Methods for Designing Industrial Process Controllers' Proceedings Colloquium on Applications of Knowledge Based Systems, London
Martin P	(1976)	'Reliability in Mechanical Design and Production' in [Henley 1976], p 267
Martin-Solis G, Andow PK & Lees FP	(1977)	'An Approach to Fault Tree Synthesis for Process Plants' Proceedings 2nd International Symposium on Loss Prevention and Safety Promotion in the Process Industries , Heidelberg, VII-367 (DECHEMA, Frankfurt)
Martin-Solis G, Andow PK & Lees FP	(1982)	'Fault Tree Synthesis for Design and Real-Time Applications' Trans IChemE Vol 60
Matsumoto K, Sakaguchi T & Wake T	(1985)	'Fault Diagnosis of a Power System Based on a Description of the Structure and Function of the Relay System' Expert Systems, Vol 2, No 3, July 1985
Maund JK	(1982)	'The Cost of Safety' Transactions 7th International Cost Engineering Congress London, p A8-1 to A8-10
McCarthy J	(1968)	'Programs With Common Sense' in: Minsky M (ed), Semantic Information Processing
McCarthy J	(1978)	'History of LISP' in: Wexelblat RL (ed), History of Programming Languages Academic Press, New York

McDermott D (1982) 'A Temporal Logic For Reasoning about Processes and Plans'
Cognitive Science, Vol 6, p 101

Melle WA (1979) 'A Domain Independent Production Rule System for Consultation Programs'
in: Proceedings 6th International Joint Conference on Artificial Intelligence, p 923

Menzies RM & Strong R (1979) 'Some Methods of Loss Prevention'
The Chemical Engineer, March 1979, p 151

Methlie LB & Sprague RH (1985) Knowledge Representation for Decision Support Systems
North Holland, Amsterdam

Michalski RS & Chilausky RL (1980) 'Knowledge Acquisition by Encoding Expert Rules Versus Computer Induction from Examples:Case Study Involving Soybean Pathology'
International Journal of Man Machine Studies, Vol 12, p 63

Michie D (ed) (1979) Expert Systems in the Microelectronic Age
Edinburgh University Press

Michie D (1982) 'High Road and Low Road Programs'
AI Magazine, Vol 3, No 1, p 21

Michie D (ed) (1983) Introductory Readings in Expert Systems
Gordon and Breach, New York

Minsky M (1975) 'A Framework for Representing Knowledge'
in: Winston PH (ed), The Psychology of Computer Vision

Miller GA, Galanter E & Pribram KH (1960) Plans and the Structure of Behaviour
Holt, Rinehart and Winston Inc, New York

Misra KB & Rao TSM (1970) 'Reliability Analysis of Redundant Networks Using Flow Graphs'
IEEE Transactions on Reliability, Vol R-19, p 19

Mitchell TM, Steinberg LI & Shulman JS (1985) 'A Knowledge Based Approach to Design'
IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-7, No 5

Moss TR (1979) 'Plant Availability Assessment'
IMechE Journal June 1979, p 91

Munday G	(1977)	'On-Line Monitoring and Analysis of the Hazard of Chemical Plant' in Proceedings 2nd International Symposium on Loss Prevention and Safety Promotion in the Process Industries , Heidelberg, (DECHEMA, Frankfurt)
Naess L & Loeken PA	(1985)	'Graphical Input to a Process Simlator' in: PSE 85; The Use of Computers in Chemical Engineering IChemE Symposium Series 92, Pergamon, Oxford
Nakashima K &_Hattori YM	(1979)	'An Efficient Bottom-up Algorithm for Enumerating the Minimal Cut Sets of Fault Trees' IEEE Transactions on Reliability, Vol R-28, No 5
Nau DS	(1983a)	'Prospects for Process Selection Using Artificial Intelligence' Computers in Industry, Vol 4, p 253
Nau DS	(1983b)	'Expert Computer Systems' Computer, Vol 16, p 63
Nelson WR	(1982)	'REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents', Proceedings AAAI-82, p 296
Nelson WR	(1984)	Response Trees and Expert Systems for Nuclear Reactor Operations Report NUREG / CR-3631, (Idaho National Engineering Laboratory, EG & G Idaho Inc, Idaho Falls, Idaho), Feb 1984
Newell A & Simon HA	(1972)	Human Problem Solving Prentice-Hall, Englewood Cliffs, New Jersey
Nielsen DS	(1971)	The Cause-Consequence Diagram Method as a Basis for Quantitative Accident Analysis RISO-M-1374 (Danish Atomic Energy Commission, Roskilde, Denmark)
Nielsen DS	(1975)	'Use of Cause-Consequence Charts in Practical Systems Analysis' [Barlow 1975a]
Nielsen D, Platz O & Kongso HE	(1977)	Reliability Analysis of Proposed Instrument Air System RISO-M1903 (Danish Atomic Energy Commission, Roskilde, Denmark)

Niida K, et al (1985) 'Some Expert System Experiments in Process Engineering'
PSE '85: The Use of Computers in Chemical Engineering, IChemE Symposium Series No 92 (Pergamon, Oxford)

Nilsson N J (1971) Problem Solving Methods in Artificial Intelligence
McGraw-Hill, New York

Nilsson NJ (1980) Principles of Artificial Intelligence
Tioga Publishing Company, Palo Alto, California

Ng SW & Walker A (1980) 'Min-Max Chaining of Weighted Causal Assertions is Loop Free'
Proceedings 1st Annual National Conference on Artificial Intelligence, Stanford, p 105

NUREG (1975) Reactor Safety Study - An Assessment of Accident Risks in US Commercial Nuclear Power Plants
WASH-1400 (NUREG 75/014)
(US Nuclear Regulatory Commission, Washington DC)

Ohsuga S (1983) 'A New Method of Model Description - Use of Knowledge Base and Inference'
in: Bo K & Lillehagen FM (eds), CAD Systems Framework, IFIP83, North Holland, Amsterdam

Olsen JV, Lind M & Taylor JR (1978) RIKKE - A Program for Automatic Fault Tree, Cause-Consequence Diagram and Simulation Model Construction
RISO-N-28-78 (Danish Atomic Energy Commission, Roskilde, Denmark)

Olsen JV, Taylor JR & Nielsen F (1979) 'Use of Automatic Fault Tree and Cause-Consequence Analysis Methods in the Analysis of A Chlorine Evaporator'
Computers in Chemical Engineering, Case Studies in Design and Control (RISO National Laboratories, Roskilde, Denmark)

Organick E, Forsythe AI & Plummer RD (1978) Programming Language Structures
Academic Press, New York

Parnas DL & Clements PC (1986) 'A Rational Design Process: How and Why to Fake It'
IEEE Transactions on Software Engineering, Vol SE-12, No 2

Patil R, Szolovitz P & Schwartz W (1981) 'Causal Understanding of Patient Illness in Medical Diagnosis'
Proceedings International Joint Conference on Artificial Intelligence IJCAI-81, p 893

Pau LF	(1981)	Failure Diagnosis and Performance Monitoring (Marcel Dekker, New York)
Pau LF	(1986)	'Survey of Expert Systems for Fault Detection, Test Generation and Maintenance', Expert Systems, Vol 3, No 2
Peate J	(1984)	Building Human Judgement into Computer Programs Process Engineering January
Peterson JL	(1977)	'Petri Nets' Computer Surveys, Vol 9, No 3, p 224
Pollack SL	(1971)	Decision Tables: Theory and Practice (Wiley-Interscience, New York)
Powers GJ & Tompkins FC	(1974)	'Fault Tree Synthesis for Chemical Processes' AIChE Journal, Vol 20, No 2, p 376
Powers GJ, Tompkins FC & Lapp SA	(1975)	'A Safety Simulation Language for Chemical Processes: A Procedure for Fault Tree Synthesis '[Barlow 1975a]
Powers GJ & Tompkins FC	(1976)	'Computer-Aided Synthesis of Fault Trees for Complex Processing Systems' [Henley 1976]
Pratt TW	(1975)	Programming Languages: Design and Implementation Prentice-Hall, Englewood Cliffs, New Jersey
Pratt VR	(1979)	'A Mathemetician's View of Lisp' BYTE, Vol 4, No 8
Preiss K	(1980)	'Data Frame Model For Engineering Design Processes' Design Studies, Vol 1, No 4, p 231
Prugh RW	(1981)	'Application of Fault Tree Analysis' Loss Prevention, Vol 14, p 1
Rada R	(1985)	'Gradualness Facilitates Knowledge Refinement' IEEE Transactions on Pattern Analysis and Machine Intelligence Vol PAMI-7, No 5

Rasmuson DM & Marshall NH (1978) 'FATRAM - A Core Efficient Cut-Set Algorithm'
IEEE Transactions on Reliability, Vol R-27, No 4

Rasmussen J (1978a) 'Notes on Human Error Analysis and Prediction'
in Apostolakis G (ed) Synthesis and Analysis Methods for Safety and Reliability Studies (Plenum Press, London)

Rasmussen J (1978b) Notes on Diagnostic Strategies in Process Plant Environment
RISO M-1983 (Danish Atomic Energy Commission, Roskilde, Denmark)

Rasmussen J (1979a) On the Structure of Knowledge - A Morphology of Mental Models in a Man-Machine System Context
RISO M-2192 (Danish Atomic Energy Commission, Roskilde, Denmark)

Rasmussen J (1979b) 'Man as a System Component'
in Smith H (ed), Man Computer Research. Academic Press, London

Rasmussen J & Rouse WB (eds) (1981) Human Detection and Diagnosis of System Failures
(Plenum Press, New York)

Rasmussen J (1983) 'Skills, Rules, Knowledge; Signals, Signs and Symbols; and Other Distinctions in Human Performance Models'
IEEE Trans on Systems, Man and Cybernetics, Vol SMC-13, No 3

Rasmussen J (1985) 'The Role of Hierarchical Knowledge Representation in Decision Making and System Management'
IEEE Transactions on Systems, Man and Cybernetics, Vol SMC-15, No 2, p 234

Reason JT & Embrey DE (1985) Human Factors Principles Relevant to the Modelling of Human Errors in Abnormal Conditions of Nuclear and Major Hazardous Installations
EAEC ECI 1164-B7221-84-UK

Reboh R (1981) Knowledge Engineering Techniques and Tools for Expert Systems
Linkoping, Denmark

Recht JL (1966a) 'Systems Safety Analysis: Failure Mode and Effect'
National Safety News, February 1966

Recht JL (1966b) 'Systems Safety Analysis: The Fault Tree'
National Safety News, April 1966

Reddy D et al (1983) 'Knowledge Based Expert Systems for Engineering Applications'
Proceedings IEEE Conference on Systems, Man and Cybernetics

Reddy R (1976) 'Speech Recognition By Machine: A Review'
Proceedings of the IEEE, Vol 64, No 4

Reitman W (ed) (1984) Artificial Intelligence Applications for Business
Norwood NJ, Ablex

Revesman ME (1986) 'Application of a Mathematical Model of Human Decision Making for Human Computer Communication'
IEEE Trans on Systems, Man and Cybernetics, Vol SMC-16, No 1

Rieger C & Grinberg M (1977) 'The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms'
Proceedings 5th Joint Conference on Artificial Intelligence, p 250

Rieger C & Stanfill C (1980) 'Real Time Causal Monitors for Complex Physical Sites'
Proceedings 1st National Conference on Artificial Intelligence
Stanford, p 215

RIJNMOND (1982) Rijnmond Public Authority
Risk Analysis of Six Potentially Hazardous Industrial Sites in the Rijnmond Area, A Pilot Study
(D Riedel Publishing, Dodrecht, Holland)

Rivas JR & Rudd DF (1974) 'Computer-Aided Safety Interlock Systems'
AIChE Journal, Vol 20, No 2, p 311

Robinson PR (1987) Using Turbo Prolog
Osborne McGraw-Hill, Berkely, California

Rosenbloom PS (1985) 'R1-SOAR: An Experiment in Knowledge Intensive Programming in a Problem Solving Architecture'
IEEE Transactions on Pattern Analysis and Machine Intelligence
Vol PAMI-7, No 5

Rosenthal A (1980) 'Decomposition Methods for Fault Tree Analysis'
IEEE Transactions on Reliability, Vol, R-29, No 2

Rouse WB (ed) (1984) Advances in Man-Machine Systems Research Volume 1
Greenwich CT: JAI Press

Rychener MD (1985) 'Expert Systems for Engineering Design'
Expert Systems, Vol 2, No 1

Sacerdoti ED (1974) 'Planning in a Hierarchy of Abstraction Spaces'
Artificial Intelligence, Vol 5, No 2, p 115

Salem SL, Apostolakis G & Okrent D (1975) 'On the Automatic Construction of Fault Trees'
Transactions of the American Nuclear Society, 22:475

Salem SL, Apostolakis G & Okrent D (1977) 'A New Methodology for The Computer-Aided Construction of Fault Trees'
Annals of Nuclear Energy, Vol 4, p 417

Salem SL, Wu JS & Apostolakis G (1979) 'Decision Table Development and Application to the Construction of Fault Trees'
Nuclear Technology, Vol 42, January 1979, p51

Sathi A, Fox MS & Greenberg M (1985) 'Representation of Activity Knowledge for Project Management'
IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol PAMI-7, No 5

Schank R & Abelson R (1977) Scripts, Plans, Goals and Understanding
Lawrence Earlbaum, Hillsdale, New Jersey

Schubert LK (1976) 'Extending the Expressive Power of Semantic Networks'
Artificial Intelligence, Vol7, p 163

Sedgewick R (1983) Algorithms
Addison-Wesley, New York

Shafaghi A, Andow PK & Lees FP (1984) 'Fault Tree Synthesis Based on Control Loop Structure'
Chemical Engineering Research and Design , Vol 62, March 1984

Shapiro SC (1979) Techniques of Artificial Intelligence
Van Nostrand, New York

Shindo A & Umeda T (1977) 'A Systems Approach to the Safety Design of Chemical Processing Systems'
in Proceedings 2nd International Symposium on Loss Prevention and Safety Promotion in the Process Industries , Heidelberg, (DECHEMA, Frankfurt)

Shiozaki J et al (1985) 'Fault Diagnosis of Chemical Processes by use of Signed Directed Graphs. Extension to Five-Range Patterns of Abnormality'
International Chemical Engineering Vol 25, No 4, p 651

Shirley M & Davis S (1983) 'Generating Distinguishing Tests Based on Heirarchical Models and Symptom Information'
Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers, New York

Shooman ML (1976) Software Reliability: Analysis and Prediction
in: Henley EJ & Lynn JW (eds) NATO Advanced Study Institute on Generic Techniques in System Reliability Assessment
Noordhoff, Amsterdam

Shubin H & Ulrich J (1982) 'IDT: An Intelligent Diagnostic Tool',
Proceedings AAAI -82, p290

Siirola JJ, Powers GJ & Rudd DF (1971) 'System Synthesis Part III: Toward a System Concept Generator'
AIChEJ, Vol 17, p 677

Silverman BG & Tsolakis AG (1985) 'Expert Fault Diagnosis under Human Reporting Bias',
IEEE Transactions on Reliability, Vol R-34, No 4, p 366

Simmons MK (1984) 'Artificial Intelligence for Engineering Design'
Computer Aided Engineering (GB), Vol 1, No 3, p 75

Simon HA (1973) 'The Structure of Ill-Structured Problems'
Artificial Intelligence, Vol 4, p 145

Simon HA (1983) 'Search and Reasoning in Problem Solving'
Artificial Intelligence, Vol 21, p7

Smith DE & Clayton JE (1980) 'A Frame Based Production System Architecture'
Proceedings 1st NCAI, Stanford

Smith HT & Green TRG	(1980)	Human Interaction With Computers Academic Press, London
Solomon CH	(1983)	The EXXON Chemicals Method for Identifying Potential Process Hazards IChemE Loss Prevention Bulletin No 52, Aug 1983
Sowa JF	(1976)	'Conceptual Graphs for a Database Interface' IBM Journal of Research and Development, July
Sowa JF	(1984)	Conceptual Structures Addison-Wesley, Reading, Massachussetts
Sowa M, Scott AC & Shortliffe EH (eds)	(1982)	'An Approach to Verifying Completeness and Consistence in a Rule Based Expert System' AAAI Magazine, Vol3, No 4, p 16
Sowizral HA & Kipps JR	(1986)	Rosie: A Programming Environment for Expert Systems in: Klahr P & Waterman DA (eds) Expert Systems: Techniques, Tools and Applications, Addison-Wesley, New York
SRC	(1983)	Science Research Council, Artificial Intelligence, A Paper Symposium
Sriram D	(1984)	'A Bibliography on Knowledge Based Expert Systems in Engineering' SIGART Newsletter, July
Steele GL	(1984)	Common LISP Reference Manual Digital Press, Bedford, Massachusetts
Stefik M	(1979)	'An Examination of a Frame Structured Representation System' Proceedings 6th International Joint Conference on Artificial Intelligence, Tokyo
Steinberg L & Kelly VE	(1982)	'The CRITTER System: Analysing Digital Circuits by Propagating Behaviours and Specifications' Proceedings of the 2nd NCAI, Pittsburgh, p 284
Stepp RE & Michalski RS	(1986)	'Conceptual Clustering of Structured Objects: A Goal Oriented Approach' Artificial Intelligence, Vol 28, p43

Struthers A & Ponton JW	(1985)	'Application of AI Techniques in Chemical Engineering' IChemE Symposium Series No 94, p 3.59
Su YL & Govindaraj T	(1986)	'Fault Diagnosis in a Large Dynamic System: Experiments on a Training Simulator', IEEE Transactions on Systems Man and Cybernetics, Vol SMC-16, No 1, p 129
Sussman G & Steele GL	(1980)	'CONSTRAINTS - A Language for Expressing Almost-Hierarchical Descriptions' Artificial Intelligence, Vol 14, p1
Taylor JH, Frederick DK & James JR	(1984)	'An Expert System Scenario for Computer Aided Control Engineering' Proceedings American Control Conference, San Diego, California
Taylor JR	(1973)	A Formalization of Failure Mode Analysis of Control Systems RISO-M1654 (Danish Atomic Energy Commission, Roskilde, Denmark)
Taylor JR & Hollo E	(1977)	'Experience With Algorithms for Automatic Failure Analysis', in [Fussell 1977]
Taylor JR	(1978)	Completeness of Automatic Methods for Failure Analysis RISO Electronics Department, (Danish Atomic Energy Commission, Roskilde, Denmark)
Taylor JR	(1981a)	Risk Analysis of a Distillation Plant Interim Report, Scandinavian Cooperation on Risk Analysis Technology, Project 20
Taylor JR	(1981b)	Fault Tree and Cause-Consequence Diagram Construction - A Compendium of Examples RISO-M-2307 (Danish Atomic Energy Commission, Roskilde, Denmark)
Taylor JR	(1982)	'An Algorithm for Fault Tree Construction' IEEE Transactions on Reliability, Vol R-31, No 2
Tillman FA, Hwang CL & Kuo W	(1980)	'System Effectiveness Models: An Annotated Bibliography' IEEE Transactions on Reliability, Vol R-29, No 4, p 295

Toller W (1985) 'The Design of Design Software'
in: PSE 85: The Use of Computers in Chemical Engineering
IChemE Symposium Series 92, Pergamon, Oxford

Touretzky DS (1984) LISP - A Gentle Introduction to Symbolic Computation
Harper and Row, New York

Tsuge Y et al (1985a) 'Feasibility study of a Fault Diagnosis System for Chemical Plants'
International Chemical Engineering, Vol 25, No 4, p 660

Tsuge Y et al (1985b) 'Fault Diagnosis Algorithms based on the Signed Directed Graph and its Modifications'
in PSE '85: The Use of Computers in Chemical Engineering, IChemE Symposium Series No 92 (Pergamon, Oxford)

Ulerich NH (1988) 'On-Line Hazard Aversion and Fault Diagnosis in Chemical Processes: the Digraph and Fault Tree Method'
IEEE Transactions on Reliability, Vol R-37, No 2, p 171

Ullman JD (1982) Principles of Database Systems
Computer Science Press, Rockville, Maryland

Underwood WE (1982) 'A CSA Model Based Nuclear Power Plant Consultant'
Proceedings AAAI -82, p 302

Vilain MB (1982) 'A System for Reasoning about Time'
Proceedings National Conference on Artificial Intelligence
AAAI-82, p 18

Wade J & Shubin H (1982) 'A Generalized Approach to Diagnostic Problems',
Expert Systems 82, BCS SGES, England

Wagner DP, Cate CL & Fussell JB (1977) 'Common Cause Failure Analysis Methodology for Complex Systems'
in [Fussell 1977], p 289

Warren DHD & Pereira L (1977) 'PROLOG: The Language and its Implementation Compared With LISP'
Proceedings Symposium on Artificial Intelligence and Programming Languages, Rochester, NY, ACM Sigplan Notices, Vol 12, No 8

Waterman DA (1971) 'Protocol Analysis as a Task for Artificial Intelligence'
Artificial Intelligence, Vol2, p 285

Waterman DA & Hayes-Roth F (eds) (1978) Pattern Directed Inference Systems
Academic Press, New York

Waterman DA (1986) A Guide to Expert Systems
Addison-Wesley, Reading, Massachusetts

Weiss SM & Kulikowski CA (1983) A Practical Guide to Designing Expert Systems
Chapman and Hall, London

Wells GL & Seagrave CJ (1979) Flowsheeting for Safety
Institute of Chemical Engineers, Warwickshire,UK

Whalley SP & Maund JK (1986) 'The Influence of Performance Shaping Factors on Operator Interactions with Process Control'
in Willumeit HP (ed) Human Decision Making and Manual Control,
Elsevier BV, Amsterdam

Whalley SP (1987) Factors Affecting Human Reliability in the Chemical Process Industry
PhD Thesis, Aston University, Birmingham

Wheeler DB et al (1977) 'Fault Tree Analysis Using Bit Manipulation'
IEEE Transactions on Reliability, Vol R-26, No 2

Whitehouse SB (1982) 'Risk Assessment: A Positive Influence on the Safety and Public Acceptability of Process Plant'
Process Economics International, Vol 3, No 1, p 51

Wightman EJ (1972) Instrumentation in Process Control
(Butterworth, London)

Williams G (1985) 'Debugging Techniques'
BYTE, June, p 279

Williams TL, Orgren PJ & Smith CL (1983) 'Diagnosis of Multiple Faults in a Nationwide Communications Network',
Proceedings 8th International Joint Conference on Artificial Intelligence, p 179

Winston PH (ed) (1975) The Psychology of Computer Vision
McGraw-Hill, New York

Winston PH & Brown RH (eds)	(1979)	Artificial Intelligence, an MIT Perspective, Volumes 1 & 2 MIT Press, Cambridge, Massachusetts
Winston PH & Horn BKP	(1984a)	LISP Addison Wesley, Reading, Massachusetts
Winston PH	(1984b)	Artificial Intelligence 2nd Edition, Addison-Wesley, Reading, Massachusetts
Winston PH & Prendergast K (eds)	(1984c)	The AI Business: The Commercial Uses of Artificial Intelligence The MIT Press, Cambridge, Massachusetts
Woods WA	(1975)	'What's in a Link: Foundations for Semantic Networks' in: Bobrow D (ed), Representation and Understanding Academic Press, New York
Woods WA	(1983)	'What's Important About Knowledge Representation' IEEE Computer, October
Wright JM	(1983)	SRL: Schema Representation Language Robotics Institute Technical Report, Carnegie-Mellon University
Wu JS, Salem SL & Apostolakis G	(1977)	'The Use of Decision Tables in the Systematic Construction of Fault Trees' in [Fussell 1977]
Yau SS & Tsai JJ	(1986)	'A Survey of Software Design Techniques' IEEE Transactions on Software Engineering, Vol SE-12, No 6, p 713
Yellman TW	(1979)	'Comment on Computer-Aided Synthesis of Fault Trees' IEEE Transactions on Reliability, Vol R-28, p 10
Young J	(1975)	'Using the Fault Tree Analysis Technique' in [Barlow 1975a]
Zakrzewski A	(1984)	'Successful Commissioning of Computer Controlled Plant' Process Engineering, Oct 1984, p 34

APPENDICES

	TITLE	Page
APPENDIX 1 :	Ammonia Let Down Case Study Paper	249
APPENDIX 2 :	Equipment Failure Mode Tables	262
APPENDIX 3 :	Example Case Study Knowledge Base Rules	265

APPENDIX ONE

Ammonia Let Down Case Study Paper

APPENDIX TWO

Equipment Failure Mode Tables

TABLE 4.2 - EQUIPMENT FAILURE MODES INDICATED BY A SINGLE INDEX NUMBER OR TWO LETTERS IN BRACKETS.

Equipment Type -	Index Number			Letters and their meanings
	0	-1	1	
Alarm	Failed			(FD) Failed to danger (IG) Ignored
Controller	No signal	Set too low	Set too high	
Control Loop	One or more valves closed	Giving Less flow	Giving more flow	
Level switch		Set low or stuck high	Set high or stuck low	(FD) Failed to danger (FS) Failed safe
Line	Fully blocked [illegible]	Partly blocked [illegible]	Fracture	(BV) Blocked valves (RV) Restricted valves
Pneumatic trip valve (3 way)	Vent branch isolated	Leaking to vent	Open to vent	
Switch (not level)		Set low	Set high	(FD) Failed to danger (FS) Failed safe
Transmitter	No signal	Indicating too low	Indicating too high	
Valve	Closed or blocked [illegible]	Insufficiently open or passing	Open or open too much	

OS [illegible]

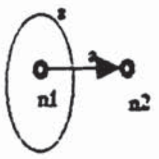
Table 3 - Code Numbers or Letters used in CAFOS To Record Failure of Equipment

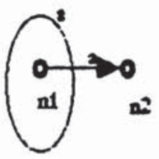
Equipment	Failure Mode Code			
Type	0	-1	1	Letters
Alarm	Failed		Operates	IG: Ignored
Controller	No signal	Set Low	Set high	
Control Loop	Valves closed	Less flow in line controlled	More flow in line controlled	
ilter	Fully blocked	Partly Blocked		L: Leaking
Flowmeter	Blocked	Indicating too low	Indicating too high	
Heat Exchanger	Tubes fully blocked	Tubes partly blocked		L: Leaking
Indicator	No signal	Indicating too low	Indicating too High	
Level Switch		Set Low or Stuck High	Set High or Stuck Low	
Line	Fully blocked with Debris	Partly Blocked with Debris	Full flow	B: Blocked by valves or fittings R: Restricted by above L: Leaking
Motor	Stopped	Running Slowly	Running Fast	
Orifice Plate	Blocked	Orifice too large or Low Density	Orifice too small or High Density	
Pneumatic Trip Valve (3 way)	Does not Vent	Leaking to vent	Vent open	
Pump	Stopped	Cavitating or low flow	Running	L: Leaking M: Motor failed E: Electrical power failed T: Power tripped
Transmitter	No signal	Indicating too low flow	Indicating too high	
Valve	Closed or blocked	Insufficiently open or passing	Open or Open too much	L: Non-return valve not seating

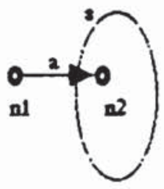
APPENDIX THREE

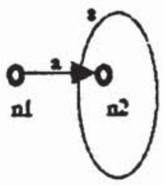
Example Case Study Knowledge Base Rules

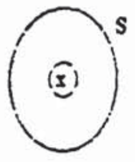
d-110-111a

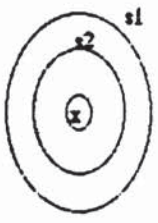
drule110		<u>ARCLEAVESSYSTEM</u> (a leaves-system := s)

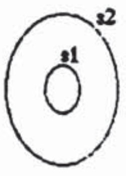
drule110a		<u>OUTGOINGARCS</u>
		
		IF (a leaves-system := s)
		
		THEN (s arcs outgoing := a)

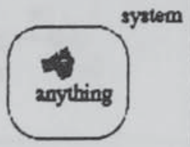
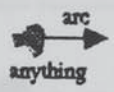
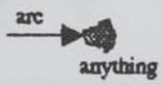
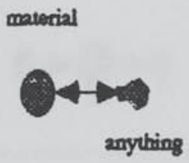
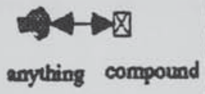
drule111		<u>ARCENTERSSYSTEM</u> (a enters-system := s)

drule111a		<u>INCOMINGARCS</u>
		
		IF (a enters-system := s)
		
		THEN (s arcs incoming := a)

drule-sh-400		SUPERSYSTEMDECLARATION
		
		IF (s system-of := x)
		
		THEN (s supersystem-of := x)
			d-sh-400-499
			
			22/10/86

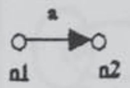
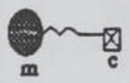
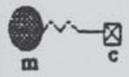
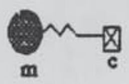
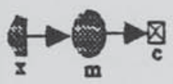
drule-sh-401		SUPERSYSTEMTRANSITIVITY
		
		IF (s1 supersystem-of := s2)
		(s2 supersystem-of := x)
		
		THEN (s1 supersystem-of := x)	

drule-sh-402		SUBSYSTEM=SUPERSYSTEMRECIPROCAL
		
		IF (s1 type := system)
		(s2 supersystem-of := s1)
		
		THEN (s1 subsystem-of := s2)	

drule1 drule2		RECIPROCITY drule descriptions 0-99 system-of -> has system has-system -> system-of relationship reciprocity
drule3 drule4		source-of -> has source has-source -> source of relationship reciprocity
drule5 drule6		target-of -> has-target has-target -> target-of relationship reciprocity
drule7 drule8		material-of -> has-material has-material -> material-of relationship reciprocity
drule9		has-compound -> compound-of

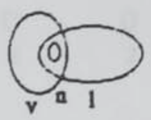
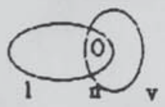
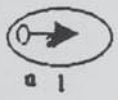
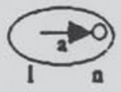
RELATOR ABSTRACT

D 100-199
drule
desc
100-104a

drule100		n1 upstream-of := n2 n2 downstream-of n1
drule101		IF n1 upstream-of := n2 n2 upstream-of := n3 THEN n1 upstream-of := n3 n3 downstream-of := n1
drule102		HAS-COMPOUNDFROMPHASELIQUID m has-compound := c
drule103		HAS-COMPOUNDFROMPHASEGAS m has-compound := c
drule104		HAS-COMPOUNDFROMPHASELIQUIDSOLUTION m has-compound := c
drule 104a		OBJECTHAS-COMPOUND $(x \text{ has-material} := m)$ $(m \text{ has-compound} := c) \longrightarrow (x \text{ has-compound} := c)$

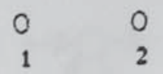
RELATOR ABSTRACT

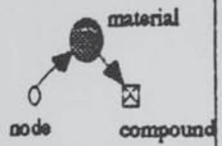
D 100-199
drule
desc
105-109

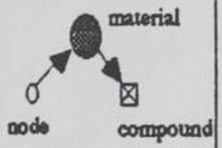
drule105		<u>INFLOWBOUNDARYABSTRACTION</u> IF (v boundary outflow := n) (l boundary inflow := n)
drule106		<u>OUTFLOWBOUNDARYABSTRACTION</u> IF (v boundary inflow := n) (l boundary outflow := n)
drule 107		<u>INFLOWBOUNDARYABSTRACTION</u> IF (PID boundary inflow := n) (l boundary inflow := n)
drule108		<u>OUTFLOWBOUNDARYABSTRACTION</u> IF (PID boundary outflow := n) (l boundary outflow := n)

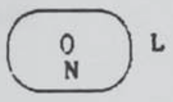
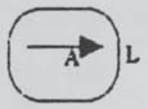
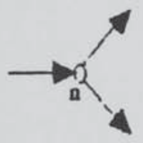
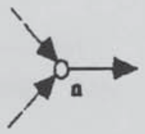
DESCRIPTOR ABSTRACT

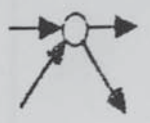
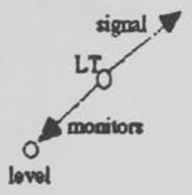
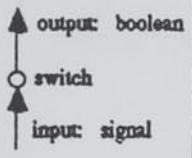
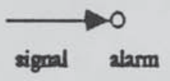
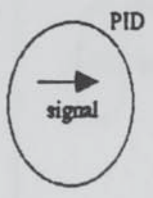
D 200-299
drule descriptions 200-299

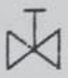
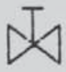
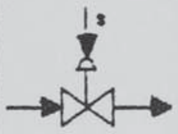
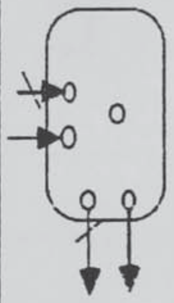
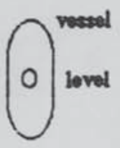
drule200		MONITOREDVARIABLE
		IF • 2 function monitor := 1
		THEN • 1 identity := controlled-variable

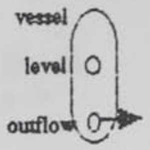
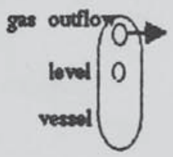
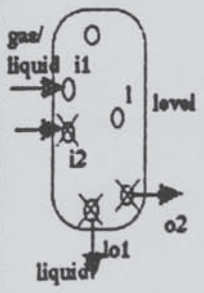
drule201		LIQUIDLINEFLOWABSTRACTION
		IF • compound is PHASE LIQUID
		THEN • (node identity := liquid)

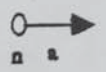
drule202		GASLINEFLOWABSTRACTION
		IF • compound is PHASE GAS
		THEN • (node identity := gas)

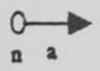
drule203		NODELINEFLOWABSTRACTION node N identity := line flow, if in line system L
drule204		ARCLINEFLOWABSTRACTION arc A identity := line flow, if in line system L
drule205		JUNCTIONABSTRACTION (n identity := junction)
drule206		JUNCTIONABSTRACTION (n identity := junction)

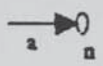
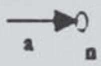
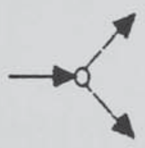
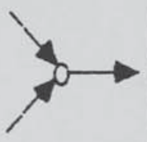
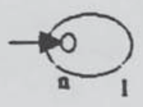
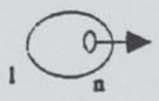
crule-c-0		CONTROLLER:INPUT/OUTPUT/FAILUREMODES crule control 0-99 simple controller with one input signal and one output signal. [+ve transfer = sign in sign out]
crule-c-1		LEVELTRANSDUCER Relationships between LEVEL, SIGNAL, and state or failure modes of Level Transducer.
crule-c-2		SWITCH high/low Causality: • Input Signal (high, normal, low, no) • Switch State or Failure Modes • Output (On, Off)
crule-c-3		alarm[alert] = alarm[failed safe]+signal[on]*alarm[normal] alarm[no alert] = alarm[failed to danger] + signal[off]
crule-c-4		LOSSOFINSTRUMENTAIR signal (no) = PID (loss-of-instrument-air)

prule-mf-l-100		OPENVALVE : Failure Modes --> Flow Deviations at valve crule mass flow 100-104
prule-mf-l-101		CLOSEDVALVE : Failure Modes --> Flow Deviations at valve
prule-mf-l-c-102		CONTROLVALVE Air to open Input Signal --> Valve Position
crule-mf-v-103		MASS BALANCE : LIQUID LEVEL IN VESSEL State of : • inflow rate / • outflow rate → • level [no less normal more] / [decreasing normal increasing]
crule-mf-v-104		LEVEL DECREASING --> NO level decreasing --> level no

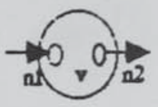
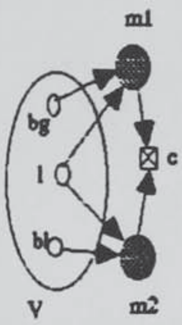
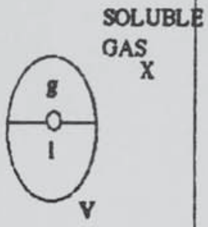
crule-mf-v-105		NO LEVEL –> OUTFLOW GAS CONTAMINATION outflow[flow as/well/as gas] = level[level no] crule mass flow 105-109
crule-mf-v-108		LEVEL INCREASING –> GAS OUTFLOW CONTAMINATION level[level increasing] ▶ gas outflow[flow as/well/as liquid]
crule-mf-v-109	 gas/ liquid i1 1 level i2 o2 lo1 liquid	MASS BALANCE: 2 PHASE SPLITTER Symptoms of • g/l inflow flow no/less/more Symptoms of • g/l inflow flow no/less/more gas/liquid

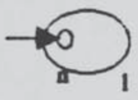
crule-mf-I-110		<u>LINEFLOWPROPAGATION</u> n flow more pd --> flow more pd a flow more pu --> flow more pu crule mass flow 110-114

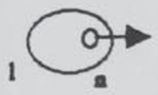
crule-mf-I-110a		IF (a id := flow normal
		n(flow no/less pd) --> a(flow no/less pd)
		a(flow no/less pu) --> n(flow /no/less pu)

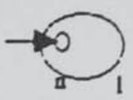
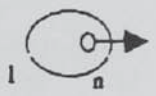
crule-mf-I-111		<u>LINEFLOWPROPAGATION</u> a flow more pd --> flow more pd n flow more pu --> flow more pu
crule-mf-I-111a		IF (a id := flow normal) a flow no/less pd --> flow no/less pd n flow no/less pu --> flow no/less pu
crule-mf-I-112		<u>JUNCTIONLINEFLOWPROPAGATION</u>
crule-mf-I-113		<u>JUNCTIONLINEFLOWPROPAGATION</u>
crule-mf-I-114		<u>LINEBOUNDARY INFLOW: CONCCHANGEPROPAGATION</u> n(conc no/less/more c) = l(conc no/less/more c)
crule-mf-I-114a		<u>LINEBOUNDARYOUTFLOW: CONCCHANGEPROPAGATION</u> l(conc no/less/more c) = n(conc no/less/more c)

crule mass flow 115-119

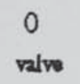
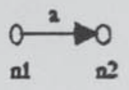
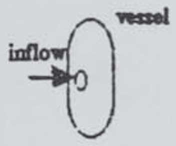
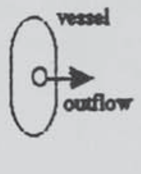
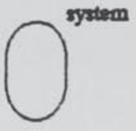
crule-mf-v-115		VESSELCONCENTRATIONCHANGEPROPAGATION n2(conc less/more c) = n1(conc less/more c)
crule-mf-v-116		VLE g(conc more c) = V(temp more) + V(pressure less pu) + V(pressure less pd) g(conc less c) = V(temp less) + V(pressure more pu) + V(pressure more pd
crule-mf-v-117		SOLUBLEGASEQUILIBRIUM l(conc more X) = V(temp less) + V(pressure more pu) + V(pressure more pd) l(conc less X) = V(temp more) + V(pressure less pu) + V(pressure less pd)

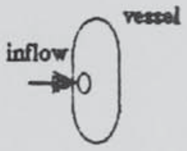
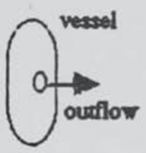
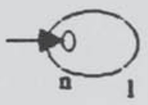
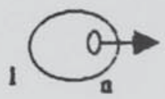
crule-mf-l-118		LINEBOUNDARYINFLOWAS/WELL/ASGAS
		
		IF (not (n id := gas))
		
		n(flow as/well/as gas) = l(flow as/well/as gas)

crule-mf-l-119		LINEBOUNDARYOUTFLOWAS/WELL/ASGAS
		
		IF (not (n id := gas))
		
		l(flow as/well/as gas) = n(flow as/well/as gas)

crule-mf-l-120		<u>LINEBOUNDARYINFLOWAS/WELL/ASLIQUID</u> n(flow as/well/as liquid) = l(flow as/well/as liquid) crule mass flow 120-124
crule-mf-l-121		<u>LINEBOUNDARYOUTFLOWAS/WELL/ASLIQUID</u> l(flow as/well/as liquid) = n(flow as/well/as liquid)

crule
pressure
300-399

crule-p-l-300		<u>PRESSURERISE DOWNSTREAMOFREDUCING VALVEONFLOW AS/WELL/ASGAS</u> valve[flow as/well/as gas] → valve[pressure more propagating downstream] (valve function := reduce-pressure)
crule-p-l-301		<u>PRESSURECHANGEPROPAGATINGDOWNSTREAM</u> n1[pressure less/more propagating downstream] → n2[pressure less/more propagating downstream]
crule-p-v-302		<u>PRESSURECHANGEPROPAGATINGDOWNSTREAMINTOVESSEL</u> inflow[pressure less/more propagating downstream] → vessel[pressure increasing/decreasing propagating downstream]
crule-p-v-303		<u>PRESSURECHANGEPROPAGATINGDOWNSTREAMOUTOF VESSEL</u> vessel[pressure less/more propagating downstream] → outflow[pressure less/more propagating downstream]
crule-p-v-304		<u>VESSEL PRESSURE: CHANGING —> LIMIT</u> Pressure Decreasing→ Pressure Less Pressure Increasing→ Pressure More

		crule temp 200-299
crule-t-v-201		TEMPERATUREPROPAGATIONINTOVESSEL inflow[temperature more/less] $\longrightarrow$ vessel[temperature more/less]
crule-t-v-202		TEMPERATURECHANGEPROPAGATIONOUT OF VESSEL vessel[temperature less/more] $\longrightarrow$ outflow[temperature less/more]
crule-t-203		LINEBOUNDARYINFLOWTEMPERATUREPROPAGATION n(temp more) = l(temp more)
crule-t-204		LINEBOUNDARYOUTFLOWTEMPPROPAGATION l(temp more) = n(temp more)