

**QUESTION ANSWERING FOR THE GENERATION OF
EXPLANATION IN A KNOWLEDGE-BASED SYSTEM**

Thesis submitted in accordance
with the requirements of the
University of Liverpool for the
degree of Doctor of Philosophy
by **Sheila Hughes.**

July 1986

QUESTION ANSWERING FOR THE GENERATION OF EXPLANATION IN
A KNOWLEDGE-BASED SYSTEM

CONTENTS	Page
ABSTRACT	1
ACKNOWLEDGEMENTS	2
1 INTRODUCTION	
1.1 Overview of Research	3
1.2 The Domain	6
1.3 Jargon	8
1.4 Overview of the Thesis	9
2 REVIEW OF THE LITERATURE	
2.1 Introduction	13
2.2 The Importance of Explanation	14
2.3 On Tolerating Aphasia	20
2.4 Explanation and Rule-Based Systems	23
2.5 The Birth of the Second Generation	29
2.6 Representing Understanding for Problem-Solving and Explanation	37
2.7 The Yale Approach to Natural Language Understanding	42
2.7.1 Conceptual Dependency Theory	43
2.7.2 Causal Chaining	45
2.7.3 Scripts	48
2.8 Question Answering and Causal Chains	50
2.8.1 Causal Antecedent	53
2.8.2 Goal Orientation	53
2.8.3 Enablement	54
2.8.4 Causal Consequent	54
2.8.5 Verification	54
2.8.6 Disjunction	55

2.8.7	Procedural/Instrumental	55
2.8.8	Concept Completion	55
2.8.9	Expectational	56
2.8.10	Judgmental	56
2.8.11	Quantification	56
2.8.12	Feature Specification	57
2.8.13	Requests	57
2.9	Primitives and Representations for Domain Entities	57
2.10	Conclusion	59
3	THE SCRIPT APPROACH TO CAUSAL MODELLING	
3.1	Overview	60
3.2	The Outline Account	61
3.2.1	Substrate Preparation	61
3.2.2	Coating Application	62
3.2.3	Drying and Curing	63
3.2.4	Wearing	65
3.3	The Script	66
3.4	Active Images	66
3.5	Stative Images	70
3.5.1	Schank's Stative Images and the Cognitive Psychology View	70
3.5.2	Applicability to the Anti-Corrosive Coating Domain	72
3.6	Causal Chaining	75
3.7	Conclusion	78
4	THE REPRESENTATION OF PROBLEM-SOLVING STRATEGY	
4.1	Overview	79
4.2	Paint Selection from Industrial Guidelines	80
4.3	Strategies for Selection	85
4.4	A Script for the Selection Task	88
4.5	Stative Images	91
4.6	Active Images	93
4.7	Classification of Script-Manipulative Tasks	94
4.7.1	The Causal/Acausal Differentiation	94

4.7.2	Differentiation of Causal Tasks	95
4.7.2.1	Basis of the Differentiation	95
4.7.2.2	Analytic Tasks	95
4.7.2.3	Predictive Tasks	96
4.7.3	Differentiation of Acausal Tasks	97
4.7.3.1	Basis of the Differentiation	97
4.7.3.2	Intraconceptual Tasks	98
4.7.3.3	Interconceptual Tasks	99
4.8	Conclusion	101
5	DESCRIPTION OF THE ADEPTUS SYSTEM ARCHITECTURE	
5.1	Introduction	103
5.2	The Knowledge Base	106
5.2.1	Overview	106
5.2.2	Scripts	107
5.2.2.1	Structure of the scripts	107
5.2.2.2	Active Images	111
5.2.2.3	Stative Images	114
5.2.2.4	The Binding List	117
5.2.3	Knowledge about coatings: The set of potential solutions	118
5.2.3.1	The coating hierarchy	118
5.2.3.2	The phase hierarchy	120
5.2.3.3	The Ideal Film	124
5.2.4	Knowledge about the Substrate	126
5.2.4.1	Substrate representation	126
5.2.5	Associated Entities	131
5.2.5.1	Liquids	131
5.2.5.2	The relation hierarchy	134
5.3	Performing the Selection task	138
5.3.1	Introduction	138
5.3.2	Situation Specific Knowledge	139
5.3.2.1	Overview	139
5.3.2.2	The substrate	140
5.3.2.3	The environment	140
5.3.3	Running \$s-script	145
5.3.3.1	Initial Instantiation	145

5.3.3.2	The MERGE act	147
5.3.3.3	The SHATTER act	147
5.3.3.4	The RUNEACH act	148
5.3.3.5	The COMPARE act	149
5.3.3.6	The RANK act	150
5.3.4	Running \$d-script	152
5.3.4.1	Initial Instantiation	152
5.3.4.2	The TOUCH process	152
5.3.4.3	The STICK process	153
5.3.4.4	The STATECHANGE process	154
5.3.4.5	The WEAR process	154
5.4	Summary	155
6	QUESTION ANSWERING AND KNOWLEDGE BASED SYSTEMS	
6.1	Introduction	156
6.2	A Reappraisal of Lehnert's Classification	156
6.2.1	Introduction	156
6.2.2	Modifications to individual categories	157
6.2.3	The Overall Hierarchy of Question Categories	158
6.2.4	Analysis of HOW and WHY using the revised classification	163
6.2.4.1	The WHY question	163
6.2.4.2	The HOW question	165
6.2.5	The Utility of the Question Categories	166
6.3	Question Answering in ADEPTUS	167
6.3.1	Introduction	167
6.3.2	The Goal Orientation Category	168
6.3.2.1	Questions about the problem-solving process	168
6.3.2.2	Goal Orientation questions about domain-level acts	175
6.3.3	The Judgment Category	179
6.3.3.1	Introduction	179
6.3.3.2	Answering Judgment Questions	181
6.3.4	The Expectation Question Category	183
6.3.5	The Definition Question Category	187

6.4	Summary	188
7	CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH	
7.1	Conclusions	190
7.1.1	Dialogue with Expert Systems	190
7.1.2	The Relationship between Problem-Solving and Question-Answering	193
7.1.3	Heuristics and Levels of Knowledge	195
7.2	Directions for Future Research	196
7.2.1	Question Categories	196
7.2.2	Domain Acts and Processes	197
7.2.3	Implicit Questions	198
7.3	Summary	199
Appendix I	Bibliography	200
Appendix II	Example Questions supplied by Unilever	209

ABSTRACT

One of the most characteristic features of Expert Systems is their ability to 'explain' their reasoning. The nature of explanation, however, is ill-defined. The work reported here approaches this problem using semantic representations based on those of the Conceptual Dependency Theory in the field of natural language understanding.

The domain knowledge and the problem-solving strategy are represented separately, but both use a script-based approach. A new class of action primitives for technical domains is described. Question answering is presented as a means of investigating the knowledge contained in the system, allowing the dynamic creation of explanation tailored to the user's needs.

A hierarchy of question types is proposed based on characteristic movements within causal chain structures, and the relationship between problem-solving and question answering is clarified.

ACKNOWLEDGEMENTS

Thanks to Brian Ward for his unfailing interest, constructive criticism and advice.

Thanks also to Mike Shave, Stuart Moralee and Brian Walsh for their help and encouragement, and to Jim Alty and Mike Coombs for initial guidance.

This research was supported by a Science and Engineering Research Council C.A.S.E. award, reference 81506775; and by Unilever Research, Port Sunlight Laboratory.

1 INTRODUCTION

1.1 Overview of Research

The explanation facilities of contemporary knowledge-based systems are rudimentary. The vast majority of such systems use rules or 'productions' as their sole method of representing problem-solving knowledge. The terms examined by such rules are very simple, forcing any richness of representation to lie in the inference structure. This methodology constrains discussion of their domain knowledge to an unwinding of chains of rules, which can be regarded simply as an edited execution trace. Production systems are undoubtedly good at producing solutions to the single task at which they are 'expert'. However, their usefulness is restricted by several factors, of which the most important is the ease with which humans can access the knowledge contained in the system.

Most IKBS designers create a system that can solve the problem, and only when that is achieved do they turn their attention to the problems of explanation and justification on the part of the system. But explanation and justification must be an integral part of the design of a system. There are many questions which simple problem-solving systems cannot answer, in spite of their capabilities in the domain of expertise. The classic

example of this is MYCIN, which for a long time could expertly diagnose bacterial infections of the blood, but was unable to explain what a bacterium was, or to decide whether a named organism known to the system was indeed a bacterium. It is a fundamental tenet of this thesis that 'explanation' can no longer be relegated to an afterthought in expert systems development. The scope of the problem must be broadened to the general field of question answering, and the familiar HOW and WHY questions of the typical rule-based systems are only a fraction of what can reasonably be asked.

The limitations on the communicative power of rule-based systems stem from their poverty in representation. One cannot expect an answer from a system if one cannot convey to the system the meaning of the concepts involved in the question. It is very easy when examining any rule-based system to construct interesting questions that the system cannot answer. Generally, any question that deals with real-world concepts (rather than ideas involved in the problem-solving process) is unaskable. For example, a rule-based system capable of selecting an optimal anti-corrosive coating for a substrate is likely to deal quite competently with a HOW question asking

'How did you decide that a bitumen coating is best?' .

It is extremely unlikely that the question

'How do you apply a bitumen coating?'

can be answered by the same system, unless it is dealt with

by specially included canned text. The difficulties associated with canned text are discussed in Chapter 2.

The view of explanation presented here is founded on the assertion that explanation depends on understanding. Heuristics do not represent an understanding of a domain, they provide a model of a particular problem-solving activity in a domain. Thus, while a useful solution can be obtained using such heuristics, one should not expect them to support sophisticated forms of explanation. Explanation is commonly associated with causal understanding of a field. Consider, for example, simple weather prediction. Using only high school geography, I can interpret basic meteorological data to some extent. I have heuristics that include, for example:

'If the pressure is low, then it will rain.'

When asked to explain about low barometric pressure however, I can produce very little helpful information. Although I have several interpretive heuristics, I have no model of weather processes and causative features. I know that it should be possible to explain such things in terms of areas of differing pressure, of 'fronts' and the temperature and moisture content of air, of the conditions that enable cloud formation etc., but these play no part in my heuristics. While I may have the vocabulary, I lack the necessary model of the physical meteorological processes. It is this kind of underlying causal model I advocate to support explanation in the form of answers to

explicit questions.

A body of work already exists in Artificial Intelligence that addresses itself to the problems of question answering. This work, by Wendy G. Lehnert at Yale, uses a deep representation of the semantics of its knowledge domain. In this thesis, I present a system whose fundamental representation is inspired by the ideas of Lehnert, which in turn were based on Schank's Conceptual Dependency theory. My aims in this research were twofold:

- (a) to develop a representation and control structure for a system capable of selecting an anti-corrosive coating for a particular environment

and

- (b) to design the system in such a way as to enable it to use all its knowledge in the most flexible way possible, and in particular to answer as many as possible of the known question types.

1.2 The Domain

The experimental knowledge-based system created in the course of this research is called ADEPTUS. The domain of expertise is the selection and use of anti-corrosive coatings for steel substrates. This engineering application is a common and important one. To increase the

quality of decision-making in selecting the best and most cost-efficient protective coating would be of enormous practical and economic benefit to industry.

The knowledge used in the work came from a variety of sources. Originally, a research engineer from Unilever Research, Port Sunlight, provided information on how selection is made on an empirical basis. This was, to a large extent, already codified in a document co-written by him and distributed throughout the Lever Bros. manufacturing company. This document is not public domain information, and so no parts of it can be quoted in this thesis.

After working with this knowledge source for some time, it became obvious that a domain expert with a rather different view of things would be needed for the implementation of structures capable of representing a real understanding of the physical processes in the domain. I needed the expertise of a paint chemist; one who knew how coatings behaved in different environments, and what caused that behaviour. This information was provided by specialists at International Paints, Marine Coatings Division, Gateshead.

The domain was selected by the industrial sponsors of the project for its relevance. It is, perhaps, unfortunate that the domain is not one of the more thoroughly understood areas of materials technology, but the difficulties

encountered here are likely to be representative of those encountered in many other interesting domains of engineering and applied science.

1.3 Jargon

Computing is infamous for its use of jargon; paint technology suffers just as much. Jargon can clarify or confuse. It clarifies when it increases precision or reduces prolixity; it confuses when an author coins a neologism to describe something which already has an accepted technical name. I have tried to avoid this unnecessary burden on the reader's memory as far as possible, except when I have been particularly unhappy with an existing word. The most notable example in this document is my unwillingness to adopt Schank's 'conceptualisation' term. I have, of course, used the term when discussing Schank's own work, but in describing my own research I use the term 'image' which the reader should regard as a synonym. Thus, references to 'stative images' and 'active images' will appear where the reader familiar with Conceptual Dependency Theory would expect to find 'stative conceptualisations' and 'active conceptualisations'. The change was made in the hope of improving the style of some sentences which otherwise got completely out of hand.

Other new or redefined jargon terms are introduced naturally as they occur, and I have tried to keep the paint technology jargon to a minimum.

1.4 Overview of the Thesis

A review of the literature is presented in Chapter 2. Influences on the work have been diverse, including philosophical views on the nature of explanation and causality, cognitive modelling and text understanding. Conceptual Dependency, causal chaining and scripts are described, as is Lehnert's question categorisation system.

ADEPTUS is essentially a script-based system. In Chapter 3, the domain of anti-corrosive coating usage is introduced, and the applicability of the script formalism demonstrated. An instantiated domain-level script is a causal model of the domain: this coincides with much of the work discussed in Chapter 2. Although Schank's primitives are examined in the literature review, the primitive acts required to represent physical processes in the anti-corrosion domain are discussed in Chapter 3.

In Chapter 4, the second type of script known to ADEPTUS is outlined. This 'system-level' script involves information processing acts undertaken by the system. The information

processing requires manipulation of the domain-level script in various ways. One major category of manipulative tasks concerns handling the domain script as a single chunk, disregarding its inner structure. For example, some aspects of the domain-level script will require instantiation, and copies of the script may need to be produced.

When dealing with individual pieces of the domain-level object script, different kinds of actions are required. The latter part of Chapter 4 describes the various manipulation tasks which can be carried out within a script entity.

The overall goal of the system-level script is to produce a list of possible solutions, ranked on suitability, to the problem of selecting the best coating for a particular situation. Chapter 4 examines the way this is done on paper in industry, and shows how a script-based representation can be created to cope with the problem.

The ADEPTUS system is presented in Chapter 5. The system is written in SRL, and a brief overview of this language is included. ADEPTUS is a preliminary implementation of some of the ideas described in this thesis. The scripts and the various structures which participate in them are described in Chapter 5, and examples are given.

Chapter 6 deals with question-answering in problem-solving

systems. It is concerned initially with the modification of Lehnert's question categories for use in a problem-solving environment, and subsequently with the question categories which have been implemented in ADEPTUS. The first part of this chapter depends upon the task analysis of Chapter 4. This is because the acts required to produce an answer to a query in a script-based system are themselves manipulations of the script. The refined classification in Chapter 6 is thus based on the analysis presented in Chapter 4. The HOW and WHY questions available in conventional expert systems are classified, and comparison made with questions that are commonly asked by non-expert participants in naturally-occurring consultations with an expert. In the second part of the chapter, the implementation of four question categories is discussed in detail. First, the goal orientation category is examined, which corresponds at the system level to the WHY question of simple rule-based expert systems. Secondly, judgment questions are dealt with, followed by expectation questions as the third category. Lastly, the definition category is illustrated. These four categories are representative of the four major divisions in the question hierarchy described in the early part of Chapter 6. Of the four, only the Goal Orientation category lies within the capability of conventional expert systems.

The last chapter presents the conclusions of the research. The progress of the work so far is evaluated, and the

relationships between explanation, question answering and problem solving are discussed in the light of the ideas presented in the body of the thesis. Several directions for further work in the area are suggested.

2 REVIEW OF THE LITERATURE

2.1 Introduction

In this chapter I shall look first at what 'explanation' is and how a computer system can hope to generate it. For particular systems, explanations may be unwanted: this possibility is dealt with in section 2.3. The first systems to generate dynamic explanation (contrasting with the use of canned text) were simple rule-based ones, and section 2.4 examines the limits of this approach.

Other types of system developed over the last ten years represent a new emphasis on the communicative abilities of knowledge-based systems and new approaches to design. I call these systems the second generation of expert systems and describe them in section 2.5.

Section 2.6 looks briefly at the formalisms used or recommended for improving communicative power, and in section 2.7 I examine the scripts employed by Schank and his co-workers to represent the basis of text understanding. Lehnert used Schank's formalism as her starting point for work on general question-answering. I discuss her views on question classification in section 2.8.

Section 2.9 examines the directions in which Schank's semantic primitives require augmentation to cope with representation of scientific and technical discourse.

2.2 The Importance of Explanation

What is meant by 'explanation'? Chambers Twentieth Century Dictionary (1979) defines 'explanation' as

"making plain or intelligible; unfolding and illustrating the meaning of; accounting for; the meaning or sense given to anything".

The important terms here are 'intelligible': able to be understood; and 'meaning'. This definition implies that explanation is an inescapably semantic task. The explainer must convey meaning in an intelligible way.

Understanding is an essential prerequisite to good explanation [Craik 1943, Johnson-Laird 1983]. As Johnson-Laird states:

"Explanation depends, of course, on understanding: if you do not understand something, you cannot explain it."

Craik, a philosopher of science, emphasises the need for scientists to create conceptual models in order to generate explanations of natural phenomena. In this sense, a model is a mapping of one domain on to another; the latter chosen for its ease of manipulation. Until a scientist has a model-based understanding of a phenomenon, he is not in a

position to explain it. Johnson-Laird declines to offer a definition of understanding, but suggests instead a set of criteria:

"If you know what causes a phenomenon, what results from it, how to influence, control, initiate or prevent it, how it relates to other states of affairs or how it resembles them, how to predict its onset and course, what its internal or underlying 'structure' is, then to some extent you understand it."

This is clearly the target for any system worthy of being described as 'expert' in its domain.

I suggest that explanation can be considered as the process of conveying understanding. A similar but more restricted sentiment is expressed by Jackson and Lefevre [Jackson & Lefevre 1984] when they describe an expert system's explanation as the process of communicating an understanding of what the system itself does. An interesting view of explanation as the transfer of declarative knowledge structures from the explainer to the listener is given by Cullingford et al. [Cullingford et al. 1981] They show how, in order to communicate through narrow verbal/graphical channels, the explainer must leave out aspects of the knowledge structure, which the audience must then fill in again. Sinnhuber points out that explanation of reasoning by computer systems is usually achieved by means of an execution trace of the system's performance. If understanding is to be communicated to a human, the system must reason in a way familiar - or at least not entirely alien - to the user [Sinnhuber 1985].

Human understanding of new information is

"a process by which people match what they see and hear to pre-stored groupings of actions that they have already experienced. New information is understood in terms of old information."

[Schank & Abelson 1977]

This indicates a cognitive modelling approach to the design of systems that can communicate their understanding.

In some situations, explanation is the primary task, provided without any prompting. The most common occurrence of this type is in teaching. However, even in a pedagogical environment, the student will often ask a question to elicit further or alternative explanation if the exposition seems insufficient or inappropriate. This thesis is concerned with communication with problem-solving systems. In such systems, explanation is not provided without request. Nor should it be since the primary aim is not tutorial.

In the medical domain, it is widely recognised that the provision of explanation is crucial to user acceptance of consultative systems. Clancey [Clancey 1981] states his position in a revealing parallel describing MYCIN as 'aphasic'. The term can be equally well applied to almost any of the expert systems now extant. Frost [Frost 1984] draws a clear dividing line between 'expert systems' and 'knowledge based systems'. For him, the term 'expert system' has come to be synonymous with a single-task problem solver. He suggests the use of 'knowledge based

system' to indicate a system not restricted to a single inferential task, but one which has a flexible deductive retrieval facility for access to the encoded knowledge. Most expert systems have been created with only one real criterion in mind - to 'get the answer right'. Awareness is now growing of the importance of the user interface as the essential factor in a system's acceptability. Edmonds [Edmonds 1982] sums up a viewpoint that is gaining credence when he advocates the design of the interface (his "dynamic processor") as an INITIAL requirement, rather than an afterthought having no real bearing on issues such as knowledge representation or knowledge partitioning. He points out that from the system designer's point of view, this interface is likely to be the most complex part of the system, outranking the problem-solver or "background processor".

"The 'object' to be designed is an interaction" says Johnson [Johnson 1985], who advises a design methodology working "from the outside in, rather than from the inside out". The theme of Johnson's paper is the adequate representation of a 'model of competence', emulating all aspects of the expert's behaviour. This implies a flexibility currently beyond the reach of expert systems. Problem Solving Is Not Enough [Cavell 1915].

From a psychological perspective, Goguen et al. [Goguen et al. 1983] point out that

"without ... the ability to interact, the speaker may present an explanation that is incoherent".

They argue that any theory of explanation must be based upon empirical study of actual human behaviour. Compelling evidence for the need for flexible methods of communication in the form of just such empirical analyses have been presented in various studies of interactions between users and human experts. These interactions have been described, with considerable accuracy, as 'naturally occurring expert systems' [Pollack et al. 1982]. In two such studies, work has been undertaken on analysis of protocols from radio phone-in programmes involving a problem-solving expert and a non-expert with a problem [Pollack et al, 1982; Kidd 1985]. The results suggest that, far from simply stating the problem and passively accepting the expert's solution, the non-expert is a surprisingly active participant in the dialogue. This is in agreement with the statement of Goguen et al. that the explainer and the listener need to negotiate a base from which successful explanation can take place.

The phone-in protocols show that the enquirer plays a major role in negotiating a problem definition with the expert, and very commonly has preconceived hypotheses about potential solutions. The non-expert may ask several questions. Many of these are requests for explanation: why a certain solution was not proposed, why the proposed solution should work, the effect of employing a non-recommended course of action, whether the expert has

considered all the factors that the enquirer sees as important. This 'critiquing' approach to problem-solving is usually judged relevant only to systems where the user is also a domain expert [Langlotz & Shortliffe 1983; Miller 1984]; but the studies by Kidd and Pollack et al., indicate that this is not so. For a selection task, where any proposed solution is likely to be flawed to some extent, self-critiquing by the system was proposed in [Hughes 1985]. This offers solutions to users, highlighting potential problems if the problem definition should alter in any important respect.

Coombs and Alty [Coombs & Alty 1982] report a series of experiments with computing-advisory dialogues. Interactions that resembled those of simple problem-solving systems (i.e. strongly controlled by the advisor, with little user participation beyond the supplying of raw evidence) were the ones judged 'unsatisfactory' by the users. 'Satisfactory' interactions displayed an apparent lack of structure, with both the advisor and the enquirer making substantial contributions to the dialogue. It was observed that much more information than was strictly relevant to the task solution was offered by both sides. The judgment of Coombs and Alty on this is that

"the very act of participation appeared to help develop a better 'understanding' of the structure of the problem ... the users had less difficulty in remembering and applying the solution."

They propose that expert systems should support

problem-solving rather than direct it.

This participatory approach is endorsed by Jackson and Lefebvre [Jackson & Lefebvre 1984] in their work on advice-giving systems. Such systems must produce explanation, but not necessarily in response to a direct user question. The system must infer a 'virtual question' from the user's behaviour, and act as though that question had been explicitly asked.

Work of this kind supports the view that expert systems can no longer be viewed ONLY as problem-solvers that adopt a 'system knows best' stance. The user must be seen as an integral part of the problem-solving 'team', and his or her knowledge of the situation fully exploited. A necessary (though not sufficient) condition for this is that the user be given access on demand to the essential understanding embodied in the system. This can be regarded as explanation, and the natural mode of prompting explanation (in any form) is the asking of questions [Coombs & Hughes 1982].

2.3 On Tolerating Aphasia

The work of Pollack et al., Kidd, and Coombs & Alty indicates that in consultations with a human expert, the user plays a large part in what is essentially a

mixed-initiative dialogue. It is still necessary, however, to consider whether or not ALL knowledge-based systems require a good explanation facility. Some constructors of expert systems think not but they are a small minority.

It seems that aphasic systems are acceptable for tasks which are routine (eg. XCON) or real-time (eg.VM); or in task domains where the encoded knowledge is claimed to be undisputed. I am sceptical of the latter case. Kahn [Kahn 1984] describes MUD, a system that diagnoses problems in lubricant drilling fluids. MUD is a system in which explanation facilities beyond those normally associated with the production system formalism (see section 2.4) are not required. He cites only two possible reasons for requiring explanation. First, to answer challenges to the line of reasoning employed. Second, to give extra information to users, whether for tutorial purposes or to permit systems to assist the user in situations other than the one for which they were designed. Kahn maintains that explanation is not necessary in the MUD system because the line of reasoning has never been challenged, and all the domain experts agree that the system is using the correct reasoning. He refutes the second necessity for explanation by asserting that this occurs only extremely rarely in this domain, and where it does, it can be foreseen and dealt with by the use of well-placed 'canned text' justifications.

The well-known XCON system (previously called R1) [McDermott 1981] has little need for sophisticated explanation capabilities. The reasons for this are essentially those given by Kahn for the MUD system, but the task is considerably more routine and well-understood. The system is known to work satisfactorily, so justification of its reasoning is rarely useful. XCON was designed to take over the task of VAX configuration, and it is difficult to see why anyone would want to use it for teaching purposes. Its highly specific nature also renders it unlikely that anyone would attempt to use it in a very novel situation.

Certain other systems also have no particular need for explanation facilities. For example, the VM system [Fagan et al. 1981] for real-time monitoring of patients using ventilator machines in Intensive Care Units, is not interactive. It collects data directly from electronic sensors attached to the patient and produces periodic reports for the use of the consultant. Fagan says

"Almost no dialogue will take place with clinicians when they are using the system"

but a real-time system might be enhanced by off-line explanation facilities that answered questions about the data captured over a certain period.

Such systems, however, are the exceptions rather than the rule. Most knowledge-based systems are designed to be interactive, unlike VM. They are usually concerned with

domains where experts do not entirely agree with each other, in critical domains eg. medicine, where users overtly demand evidence of acceptable reasoning, or where use of the knowledge-base for tasks other than that initially intended (eg. teaching) is advantageous. Two types of expert system will exist in the future: those that do not require sophisticated explanation facilities and those that do. The former will be constructed in a very much simpler fashion than the latter.

2.4 Explanation and Rule-Based Systems

When criticising the inflexibility of current expert system explanation facilities, it is salutary to remember how much of an advance it was to generate any explanation at all. Such capabilities are the true hallmark of knowledge-based systems, and it is just those capabilities that create the gulf between expert systems and conventional programs.

The two modes of explanation of a rule-based system are described as HOW and WHY questions. The subject of both types of enquiry is a term present in either a premise or a conclusion of a rule in the system. Both questions can in theory be asked in the general sense or with reference to a particular situation. So, if no problem-solving has taken place one may ask

'WHY would the system need to know X?'

Explanation is then provided by finding all the rules which have X as a premise term, and quoting them as possible reasons for trying to establish X. During or after problem-solving, one can ask

'WHY did the system need to know X?'

This time, only those rules having X as a premise term and which were actually fired, or which are currently under active consideration to be fired, will be quoted. The mirror image of these two questions can be constructed to answer the corresponding questions

'HOW would you prove Y?'

and

'HOW did you prove Y?'

For HOW questions, Y is matched against rule conclusion terms and rules showing such a match are quoted.

The whole basis of the production system methodology, and the claims that it represents a good model of human cognition [Davis & King 1977], are attacked by Johnson-Laird [Johnson-Laird 1984]. Johnson-Laird challenges the belief that humans commonly reason in a formal manner. He demonstrates the use of mental models in reasoning, and points out that such model-based reasoning is closer to the actual techniques employed by humans than is formal logical reasoning. Acceptance of such claims in full would force a radical revision of the role of logic in systems that attempt to emulate human cognition even in part.

Rule-based systems are rarely a representation of an understanding of the domain of expertise, and this putative understanding is not communicated to the user in a comprehensible form by quoting a chain of rules. The rule-based system typically contains rules which have a great deal of knowledge implicit in them. This is not only the tacit knowledge described by Collins et al [Collins et al. 1985], but is information concerned with the domain which may well be unfamiliar to the user. For example, Clancey [Clancey 1983] and Johnson [Johnson 1985] both quote the ordering of clauses within a rule as containing implicit knowledge. In the same paper, Clancey says that conflict resolution strategy commonly remains implicit, usually indistinguishable from the control structure of the interpreter. The conflict resolution strategy does however form a part of the overall understanding of the domain. The fact that we usually cannot access it in an explicit form means that the system essentially does not understand that part of the domain.

Production rules are highly modular in that they are self-contained units of knowledge, requiring no external context to be valid statements of a useful piece of domain information. They can be added to or removed from a rulebase without fear that they will be fired in an inappropriate situation, since the rule itself specifies the situations in which it is valid. Davis et al. [Davis et al. 1977] point out that the very modularity of the rules

means that all context information must be contained in the rule premise. The inescapable result of this is long, complicated and hard-to-understand premises for the rules.

These factors have a bearing on whether or not the user finds the quoted chains of rules intelligible. Davis et al. [Davis et al. 1977] challenge this at a very basic level. They highlight the fact that it remains an assumption that a system rule, when translated into something approaching natural language, provides a reasonable explanation of what the system is doing. Often, rules must be employed in a system which are content-free, for reasons of control. Such rules are not obtained from the domain expert, and do not represent a single piece of knowledge about problem-solving in the domain. The rulebase is being used to hold information on both domain problem-solving and the control structure of the interpreter. While this might be a positive trait if all the control information were explicitly represented in this way, it can be nothing but confusing when such control rules only occur occasionally and represent only a fraction of the structure of the interpreter. These control rules can of course be produced in an explanation chain, and can make understanding of the system's problem-solving method extremely difficult [Johnson 1985].

Some rules may have very many premises and the situation they describe may not be at all clear to the user. For

example, Coombs and Alty [Coombs & Alty 1984] point out that mixed initiative dialogue requires that the user be allowed to propose novel goals during the course of a consultation. In production system terms, this implies the enforced use of forward chaining, since by definition novel goals cannot already be present in the system's network of inferences. However, the control of forward chaining systems can be a difficult problem because combinatorial explosion of possible inferences can easily occur. Conflict resolution by means of complex meta-rules has been suggested as a suitable control in such cases, but Coombs and Alty illustrate the cognitive opacity of such meta-rules, which hinders rather than helps explanation of the system's function.

It is also easy to muddy the explanation when a rule is used in which the conclusion uses a value computed in a premise term rather than the (more intuitive) success or failure of the evaluation of premise terms [Davis et al. 1977]. The usefulness of a code paraphrase as explanation is limited by the information represented, but more importantly by the information not represented [Neches et al. 1985], i.e. the causal, strategic or context knowledge which has had to be compiled out to push the domain knowledge into a rule-based formalism.

In spite of these problems, the rule-based approach to explanation has not been without its champions. Even after

considering its drawbacks, Davis et al. [Davis et al. 1977] have claimed that the rule-based approach is sufficiently intuitive that a natural language version of an execution trace is "a reasonable basis from which to start" explanation. Davis' position has changed since he co-authored the 1977 paper. The desire for problem-solving flexibility has led him to abandon the uniform production system methodology in favour of modelling domain causality and the flow of inference. These '2nd generation' expert systems are examined further in section 2.5.

Pre-arranged or 'canned' text is the simplest form of explanation available to a problem-solving system. Canned text has occasionally been recommended as a satisfactory method of helping the user make sense of the system's behaviour and results [Kahn 1984, Chandrasekaran & Mittal 1983]. However, the maintenance of such text is a separate, additional task to the maintenance of an evolving knowledge base, and like any form of static documentation, it can easily get out of step with the true state of the system [Neches et al. 1985]. For an expert system with a knowledge base that is often changed or updated, this type of static documentation is inherently dangerous.

Several workers have attempted to improve the usefulness of the HOW and WHY facilities in rule-base systems. Swartout [Swartout 1977] in his early work, the Digitalis-Advisor system, acknowledges that to provide a good explanation the

system's reasoning must in some way correspond to the user's model of the domain processes. He attempts to improve explanation by identifying places in the computational model where it is clearly different from that of the user. At such places, Swartout used 'comments' (i.e. some form of canned text) in the code to paste over the non-intelligible part of the computational model. Digitalis-Advisor uses several levels of abstraction to control the amount of information generated during an explanation. The system offers the two forms of the HOW question: a form asking a general question (the 'how can you..?'), which Swartout calls DESCRIBE-METHOD, and the form referring to a specific case (the 'how did you..?') which he calls DESCRIBE-EVENT. Swartout's later work employs a more sophisticated view of explanation, and is described in the next section.

2.5 The Birth of the Second Generation

[Swartout 1981] deals with a system called XPLAIN. In this he employs a domain model, and uses an automatic program generator to construct a problem-solving program from the domain model by refinement from abstract goals. Explanation is achieved by examination of the refinement structure created by the automatic program generator. The aim of this information is to offer an explanation of the underlying causal basis of a particular rule, commonly

referred to as a rule justification. This extends explanations in Swartout's systems from a statement of the rules used in his 1977 paper to access to an underlying causal model of the domain in his 1981 work.

Patil, in his ABEL system [Patil 1981], employs a multi-level rule-based causal model, linked by common terms or nodes which are available to more than one level. His aim in using such a representation is, like Swartout's, to improve the explanatory power of the system, which is a diagnostic problem-solver in the domain of acid base and electrolyte disturbances. Like Swartout's XPLAIN, this facilitates justification of high level rules in terms of chains of lower level causal rules that connect nodes referenced by the premise and conclusion of the high level rule. Patil's fundamental view of diagnosis is the attempt to provide an adequate explanation of the observed findings, using 'explanation' in Craik's sense of scientific explanation. From this basis, he dismisses a set-covering model of diagnosis as unsatisfactory because of the causal nature of disease processes. He therefore selects a causal model as his knowledge representation. Causality is represented as a rule network, and the generation of a justification is a statement of a chain of these causal rules. Patil generates several versions of his Patient Specific Model, and the different models compete. Like the system of Addanki and Davis, this offers the possibility of comparing outcomes in possible worlds.

Ferrand [Ferrand 1984], in his SESAM system, combines the approaches of Swartout and Patil. He employs an automatic program generator in conjunction with a multi-level causal model of the domain.

In the work of Swartout, Patil and Ferrand, the explanation is seen as a single task produced in a uniform way regardless of the user's detailed requirements. The user is not permitted to isolate aspects of the domain that interest him; if he is interested in a rule, then he can have the causal basis for that rule; if interested in a term, he can either discover its consequent inferences or its logical antecedents. Explanation is viewed as a specific task, rather than a generic term describing the many ways in which understanding may be communicated. On this level, the problems of open user access to a real understanding of the domain are not fundamentally dealt with.

The PEA system of Neches et al. [Neches et al. 1985] offers advice on improving Lisp code, for example replacing CONDS by IF or UNLESS to improve readability. The system is a development of Swartout's XPLAIN methodology, having an expanded knowledge base on which a program generator operates. The originators of the system believe that explanation should be viewed in terms of a collection of different information goals, contrasting with the usual view of explanation as a single uniform task. Each

information goal is to be associated with a particular class of question, and has an appropriate explanation strategy associated with it. This view of explanation as a multitude of tasks is similar to mine. Lehnert [Lehnert 1978] in the field of natural language understanding has also addressed the problems of answering questions (see section 2.8). An earlier version of this question answering view of explanation was put forward by Scott et al. [Scott et al. 1977], with respect to MYCIN. They proposed that at the program design stage the question types that were to be handled by the system should be enumerated. The explanation capability of the system would then comprise a set of specialist explainers, one being provided for each question type. The underlying representation was MYCIN's production system approach, but information was also to be available describing the form in which static and dynamic (i.e. inferential) knowledge was held in the system. Clancey, one of the co-authors of the paper, in later work [Clancey 1983] shows how justifications for rules fall into four categories: justification by identification, by causal factors, by common sense, and by restatement in terms of domain-dependent facts. This is an important elaboration of the idea of specialist explainers.

In early work [Clancey 1977] Clancey demonstrates an explanation facility that is not a statement of rule chains. The system he describes is of a generate and test type, and leaves behind a static event structure as it

executes its problem-solving task. Examination of this event structure can answer two categories of question:

'Why was drug X prescribed for organism Y ?'

and

'Why wasn't drug Z prescribed for organism Y ?'

Although this is not an unwinding of rules, it is still an examination of an execution trace.

In his description of the GUIDON system, Clancey argues [Clancey 1979] that rule-based problem-solvers are fundamentally unsuitable for use in tutorial tasks. Two new levels must be added to make such a system useful for teaching. First, a support level, which provides justifications for individual rules, and second, an abstraction level, which defines the strategy for the use of problem-solving rules and organises these rules into patterns. GUIDON also has an explicit representation of teaching expertise.

The later approach to explanation espoused by Clancey [Clancey 1983] is far more broadly based. In attempting to use the MYCIN system for teaching purposes, Clancey recognised that lack of explicit representation of domain understanding was the fundamental cause of MYCIN's inability to explain itself. He reconfigured MYCIN to the system NEOMYCIN, including two new categories of rule. Rules explicitly describing diagnostic strategies are independent of the particular domain reflecting simply the

way a medical diagnostician in any field approaches the task. Secondly, causal rules were included. Clancey indicates that access to a causal model is essential for the user to understand the system's reasoning. Indeed, it has been shown [Hasling et al. 1984] that the explanation of strategy is extremely difficult unless domain-level concepts are understood first.

In MYCIN, information at various levels - causal, strategic, problem-solving - was inextricably interwoven in a single level system. In NEOMYCIN, Clancey separated out the knowledge so that each level was a coherent description of that view of the domain. NEOMYCIN can be viewed as a combination of GUIDON's explanation capabilities and MYCIN's problem-solving.

A causal network showing the progress from the initial presence of pathogenic conditions to eventual symptoms is the underlying knowledge base for CASNET [Weiss et al. 1978], which is competent in the field of glaucoma diagnosis and therapy. Such networks permit not only diagnosis but also prognosis. The nodes of each network correspond to physiological states, and the links represent physical transitions between the states. Diagnosis is achieved by comparison of the fully modified patient model with networks representing the progress of various diseases. The best match provides the offered diagnosis. Szolovits and Pauker [Szolovits & Pauker 1978] offer a

comparison of MYCIN's inference net structure with the causal net structure of CASNET. The fundamental difference between the two representations is in the semantic interpretation of the nodes and arcs in the two systems. In MYCIN, a node is an arbitrary world fact, and a link is a similarly arbitrary implication joining the facts on an entirely empirical basis. In CASNET, a node represents a physical state of the domain, and a link represents physical causality. Problem solving inferences are not intertwined with physical effects, leaving CASNET's semantics certainly more consistent and arguably clearer to a human observer.

The use of functional and structural models to represent faults in digital electronic systems is advocated by Davis et al. [Davis et al. 1982, Davis 1983]. These models are expressed as a series of commands, which, when executed, create data structures that model the components and their interconnections. The fault models must be quite distinct from any problem-solving knowledge: the 'flow of electricity' and the 'flow of inference' are quite independent.

A similar idea is presented in the PROMPT system [Addanki & Davis 1985], which addresses the problems of design of new specialised objects, e.g. a ball point pen for use in conditions of unusual temperature, gravity etc. PROMPT holds prototypes of the objects of interest (i.e. the pen)

and transforms a prototype to effect design. Prototypes are also available to describe physical processes and relations. The system can also cope with diagnosis of faults in existing instances of such objects and with prediction of the behaviour of specific objects. The description of the problem solving process is described in terms of 'states', which are snapshots of the world at a fixed time. If a full description of a state is unnecessary, a 'scene' or partial state can be used. A 'history' maps an interval of time in the real world into an ordered collection of scenes, yielding a partial view of the physical occurrences in the period. As states correspond to scenes, so 'chronicles' correspond to histories, offering a complete world description for the time interval. To achieve the various types of problem-solving required, multiple pasts and futures can be generated in the form of multiple parallel chronicles. Although not of primary importance to Addanki and Davis, the multiple chronicles will allow description by PROMPT of why a proposed solution would (or would not) work.

The systems described in this section all highlight the important point that improved explanation requires the representation of extra domain knowledge which is not directly relevant to a simple problem-solving task. A system can usually make do with a simple representation of the domain, but to understand that domain to the point where it is possible to begin to generate explanation, more

sophisticated and detailed knowledge and knowledge representations are required.

2.6 Representing Understanding for Problem-Solving and Explanation

For those systems in which understanding of the domain leads to superior explanation facilities (whether the explanation is volunteered or must be requested), what type of representation has been recommended and used? As early as 1943, Craik [Craik 1943] was advocating the use of a causal model as the basis for generating explanations of scientific phenomena. Johnson-Laird refers to mental models, which are manipulated in a simulation process within the human brain to perform inference and problem-solving. His concern is with the modelling of understanding. These observations are upheld by the views described in section 2.5. All the more successful explainers incorporate a knowledge of causality in the appropriate domain; some insist on explicit strategy representation also.

In creating the TEIRESIAS system [Davis 1977] for the interactive acquisition of knowledge from experts, Davis chose to view understanding in terms of matching incoming data against an existing model. In his case, the model was of a characteristic rule, a typical member of a subset.

The view is widespread; many workers have (implicitly or explicitly) seen the system's knowledge as a model or collection of models - of physical causality, of the expert's problem-solving processes, of general strategic knowledge - against which data from the outside world is compared and used in accordance with the expectations inherent in those models. [Addanki & Davis 1985, Aikins 1983, Chandrasekaran & Mittal 1983, Clancey 1979, 1981, 1983, Clancey & Letsinger 1981, Davis 1980A, 1980B, 1983, Davis et al. 1982, Ferrand 1984, Goguen et al. 1983, Hagert 1985, Hasling et al. 1984, Jackson & Lefevre 1984, Johnson 1985, Kidd & Cooper 1985, Koton 1985, Langlotz & Shortliffe 1983, Neches et al. 1985, Patil 1981, Swartout 1981, Weiss et al. 1978, Weld 1985, Van Releghem 1984] . Clancey [Clancey 1981] argues that the use of expert systems for tutorial purposes (i.e. for good explanation, among other things) requires a shift in viewpoint from solving problems well to a simulation of the reasoning process. In articulate problem-solving systems, we should then look to incorporate such a simulation into the system.

An interesting comparison of a rule-based system and a model-based system in the same domain was made by Koton [Koton 1985] using GENEX and GENEX II. Both are concerned with problem-solving in molecular biology. Koton reports increased problem-solving power of the model-based system, demonstrating its capability to deal with novel situations. These situations were ones which were simply not prefigured

in the rule premises and therefore insoluble to the rule-based system. The advantages of the model-based system were flexibility, the ability to change the model in a unique manner equivalent to making a collection of disjoint changes to many rules in the rule-based system, and the accessibility of the domain knowledge. However, Koton does point out that of the two systems, GENEX II, the model-based system, requires a much more complicated control structure. The tasks posed to both systems were of a predictive, 'forward-chaining' nature. To what extent the tasks affected the comparison, Koton does not say.

Many systems have used the rule-based formalism to represent these models, including NEOMYCIN, ABEL and SESAM. Others, e.g. CASNET and GUIDON have employed state-transition networks. For example in GUIDON, the event structure originating from the execution of a generate-and-test strategy can be regarded as a state-transition diagram of states of knowledge in the system, and transitions or actions leading to the next state. These constitute instantiated problem-solving methods, and as Clancey points out, are readily comprehensible by the user [Clancey 1977].

A most appealing developmental view of expert system construction is presented by Riesbeck [Riesbeck 1984]. Although a 'prototyping' approach is often advocated as a good method for building expert systems, such systems never

'learn' in any meaningful sense; knowledge may be added in a modular fashion, but its organisation remains strictly under the control of the knowledge engineer. Like Athene, systems spring fully grown from the mind of their designer; they have never been novices, always instant experts. Riesbeck challenges this methodology, listing three advantages which would be enjoyed by a system which had been 'educated' in its domain of expertise. First, non-expert knowledge (which would of course be retained by the eventual expert) could be used to explain decisions to non-expert users. Secondly, non-expert knowledge could be used to handle novel situations since it may be less effective for problem-solving, but is likely to be more general than the later-developed skill. And finally, if the system were created by 'teaching' it, continued updating of a volatile domain knowledge base would be very simple. Riesbeck suggests that novice knowledge is domain-centred, involving physical objects and causality. Expert knowledge, in contrast, is task-centred, organised around the kind of answers the expert wants to find. This leads to knowledge of relations between a situation and a solution - typically heuristic.

In representing models of understanding, work in another area of Artificial Intelligence is relevant. Charniak [Charniak 1981] points out that traditionally, frames have been used for natural language work that requires a shallow representation of a broad domain, and rules have been used

as the most common formalism for problem-solvers, where the requirement is for a deep representation of a narrow domain. He advocates a combination of the two approaches to broaden understanding in problem-solving systems. Following this line of reason, Aikins employs a combination of frames and rules. Her CENTAUR system [Aikins 1983] uses prototypes coded as frames to represent disease classes. These prototypes are hierarchically organised, and the system fills the frame slots by utilising knowledge bases in the form of rules associated with each slot. Aikins points out that to restrict oneself to the single formalism of rules and not to take advantage of frame-oriented representations is unnecessarily restrictive.

The demand for flexibility and for an explanatory capability means that the information previously held in rules must be broken down into its constituent parts. Also, knowledge which has no direct involvement in the problem-solving process needs to be stored. Frames provide representation structures which have been used in other areas to represent this wider variety of knowledge.

A group of workers at Yale have demonstrated understanding by answering questions using scripts as their underlying representation. The argument of this thesis is that this approach offers considerable promise for application to explanation in problem-solving systems. In the remaining sections of this chapter this approach is examined in more

detail. The work of Schank and Lehnert is covered in greater detail than that of other researchers discussed in this review because their ideas are of central importance to this thesis.

2.7 The Yale Approach to Natural Language Understanding

Over the last ten to fifteen years, cognitive scientists at Yale have approached the questions of understanding and memory organisation in various ways. Their methodologies have ranged from the simple (e.g. understanding simple sentences using conceptual dependency representation) to the complex (e.g. goal based understanding to explain apparently unrelated actions in social situations; memory organisation packets illustrating learning mechanisms).

But whatever the level of complexity, the emphasis has remained on achieving a deep semantic representation of the text. This work on text understanding is relevant to any attempt at semantic representation: as Schank asserts in an attack on the syntax-oriented approach to natural language understanding and generation [Schank 1981],

"the problem of text representation is identical to the problem of meaning representation".

In the work of the Yale group, this is usually based on conceptual dependency (CD) theory in some form. The

remaining sections in this chapter start with an overview of CD theory, building from there to scripts, and to question-answering by the QUALM module.

2.7.1 Conceptual Dependency Theory

The conceptual dependency theory is described by Schank in [Schank 1975]. This theory was aimed primarily at developing a representation of the meaning of individual simple sentences in natural language. The topics covered by his simple sentences were common actions performed by people. Schank, like other workers both before and since, viewed the meaning of the sentence as verb-centred and actor-oriented. He wished however to perform more than a simple case-grammar analysis, which gives the verb used in the sentence a central position and assigns all other elements to case 'slots' attached to the verb. His aim was to capture the essential 'meaning' of the sentence, defining the verbs in terms of some underlying fundamentals. To this end, he first declared an axiom and its corollary, in an attempt to define how he would measure the success or failure of his 'meaning' representations.

Axiom : Two sentences having identical meaning, regardless of language, must share a unique representation.

Corollary : Any information implicit in a sentence

must be made explicit in its
representation.

So the three sentences

'John sold the book to Jane'

'Jane bought the book from someone called John'

'Jane purchased the book from John'

would all have an identical representation, and the implicit concept of transfer of money would be explicitly shown. (Schank does however distinguish the difference in 'focus' between the active and the passive forms.) Each simple "meaning proposition" is called a conceptualisation. Active conceptualisations describe the meaning of simple action, and Schank uses a uniform syntax to represent them:

Actor Action Object Direction (Instrument)

Stative conceptualisations are simple unary or binary predicates, represented using the form:

Object (is in) State (with Value)

In aiming for a language-independent representation of meaning for his chosen domain, Schank postulated eleven 'primitive acts' in terms of which all the actions in the domain of interest could be described. For example, the primitive act for the three examples in the preceding paragraph is ATRANS, the transfer of an abstract concept of possession or control. These primitive acts could then be used in the place of the ordinary verb in a case-grammar analysis. The active conceptualisations representing each

example sentence are

(subject JOHN) (act ATRANS) (object BOOK) (to JANE)

(subject JANE) (act ATRANS) (object MONEY) (to JOHN)

Some sentences do not, however, describe an action. These are stative conceptualisations; that is, the sentences describe states rather than acts. For example:

'Jane was overjoyed'

'Jane was happy'

'Jane was sad'

'Jane was suicidal'

Schank did not propose 'primitive states' to describe such assertions, but chose rather to represent the stative conceptualisations in terms of named 'scales' ad hoc. Values on these scales range between -10 and +10. The above examples are chosen from Schank's MENTAL STATE scale.

The sentence

'Jane was overjoyed'

might thus be represented as

(subject JANE) (state MENTAL STATE) (value +9)

2.7.2 Causal Chaining

Having established a meaning representation that could adequately fulfil his CD axiom and corollary, Schank addresses the problem of the meaning representation of connected text. How can meaning be shown to be connected

between two sentence representations ? Sometimes explicit connectors are provided and a direct causal link can be identified, but often apparent links are misleading if treated literally. Consider the following two simple examples.

'It was raining. John got wet because he had no umbrella.'

Here, the rain plus the lack of umbrella are the factors that cause John to get wet.

'It was raining. John got wet because he had forgotten his umbrella.'

This cannot be taken literally: the mental act of forgetting cannot directly cause one to get wet! In this case, the act of forgetting leads to a state where John does not have an umbrella: it is this state which enables him to get wet when it rains.

From considerations like these, Schank arrives at a representation of connected text which consists of chains of causation. He describes five rules of causal syntax, each showing a means of connecting active and stative conceptualisations. The five rules are

(i) Actions can result in state changes.

ACT => STATE

r

(ii) States can enable actions.

STATE => ACT

E

(iii) States can disable actions.

STATE => ACT

dE

(iv) States or acts can initiate mental states.

STATE => MENTAL STATE

I

ACT => MENTAL STATE

I

(v) Mental states can be reasons for actions.

MENTAL STATE => ACT

R

(Note: Schank's written representation of the causal links would show the arrows pointing from right to left in statements (i) to (v) above; I find the above notation preferable.)

The letters associated with the arrows in the diagrams denote the five types of causal links:

r results in

E Enables

dE dis(En)ables

I Initiates

R Reason

Using these rules, causal chains can be built up, connecting the active and stative conceptualisations

present in the text. Where a literal interpretation of the text leads to a syntactically incorrect linkage (as in forgetting leading to getting wet), the 'missing links' of the causal chain must be filled in.

2.7.3 Scripts

Using causal chains of conceptualisations, Schank describes the representation of simple connected text where the actions could be linked together in a way determined almost completely by the explicit content of the text. However, such text has to be contrived; to use the technique on simple newspaper stories remains beyond the scope of the work at this point. The reason for this is clear. People understand, for example, stories about car accidents given in an extremely concise form because they have previous knowledge of an 'archetypal' car accident story. Consider the piece of text:

'A man was killed and another seriously injured yesterday when a car veered off the B5181 and hit a tree. An ambulance arrived at the scene within minutes, but the driver of the car was dead on arrival at the hospital. Police have not yet released the names of those involved.'

When we read the text, we are not surprised at the sudden and unexplained mention of an ambulance: it is a

well-comprehended part of our expectations about stories of this kind. If 'ambulance' were replaced by 'ice-cream van' we would be very puzzled at its inclusion in such a tale. Understanding of the text is only possible because we already have available a knowledge structure which Schank refers to as a SCRIPT. These scripts, then, play a central role in Schank's text understanding work. They are large uninstantiated causal chains.

The causal chains may have several alternative branches representing different possible paths to a particular act or state. If we consider, for example, a launderette script, then the action of putting soap powder into the machine requires that the main actor of the script must first be in possession of some soap powder. There are several ways in which this can happen: he may buy it on the spot; he may ask another launderette patron for a donation from their supply; or he may be highly organised and bring it with him to the launderette. Often, when appropriate text is read, only one of these alternative paths is instantiated. However, in some cases, no relevant information for deciding between the alternatives is present in the text, and all remain as possibilities.

These archetypal stories in the form of uninstantiated causal chains must, of course, have actors featured in them. These actor slots in the uninstantiated script are occupied by role descriptors; for example, in the car

accident story, the roles in the script may include

Car Driver

Passenger

Eye Witness

Ambulance Driver

These roles are assumed by the people mentioned in the actual text when the text is read and the script instantiated.

Schank's examples of natural language text are often entertainingly bloodthirsty, usually describing various ways of killing and maiming. Unfortunately, the domain of anti-corrosive paints does not offer such diverting examples!

2.8 Question Answering and Causal Chains

Lehnert [Lehnert 1978] recognised that understanding of text is most convincingly demonstrated by the ability to answer an assortment of questions about that text. This idea has been used for generations in primary and secondary schools, and in examinations. Lehnert constructed a module called QUALM, which could translate natural language queries into a Conceptual Dependency representation and then access the associated instantiated script to obtain an answer. This was achieved by dividing the question into two conceptual parts: the question category and the question

concept. Lehnert's question concept is the CD conceptualisation which is present in the question. This contains no interrogative force, but clearly may also have one or more slots uninstantiated.

For example, using a story about John going to London by train in order to go to the theatre, we may consider the following questions:

How did John go to London?

When did John go to London?

Why did John go to London?

Although all three example questions are clearly different and require different answers, if the interrogative force of each query is removed, we are left with a common central concept:

John went to London.

This simple sentence is readily representable in CD terms, and is the question concept for our three example questions.

The concept derived from a posed question can be matched against the story representation. When a match has been found, the mechanism for recognising an associated item which constitutes a reply depends upon the interrogative part of the question. So, having discovered a match for

John went to London

in the instantiated script story representation, we obtain answers to our three questions in different ways by

reference to the question category for each question.

How did John go to London?

requires an answer describing the mechanism employed to achieve the action, i.e. by train.

When did John go to London?

enquires about a temporal location for the act, which could either be absolute or relative to the time of enquiry. For example,

On the 22nd June 1985.

Last Friday.

could be sensible answers.

Why did John go to London?

is a rather more complex question than the previous two examples. The complexity stems from the nature of the question category. In the previous two examples, responses could be found by examining slots attached to the story conceptualisation matching the question concept. No other conceptualisation needs to be accessed. However, this question asks about John's motives in engaging in the act in question. This necessitates looking ahead in the causal chain to see what major events are subsequently enabled by the question concept. Thus, for this question category, the question-answering mechanism must shift its attention to conceptualisations in the causal chain other than that which matches the question concept.

Lehnert identifies thirteen categories of questions.

2.8.1 Causal Antecedent

e.g. Why did the glass shatter?

Questions in this category ask about states or events that have in some way caused the question concept. They are answered by identifying the prior concept in the instantiated causal chain.

2.8.2 Goal Orientation

e.g. Why did John go to London?

These questions ask about the goals that prompt an action.

Lehnert says

"this presupposes that the actor of the question concept is a human who acts of his own volition".

Goal orientation questions are meaningless outside of the context of a sentient actor. The category differs from causal antecedence in that here the action (goal) described in the answer has not yet happened; replies to a causal antecedent question are found in prior states or acts, not subsequent hypothesised acts. Clearly, goal orientation questions are closely related to causal antecedent questions, but Lehnert does not examine the nature of this relationship. In analysing the mechanisms for question answering implemented in QUALM, she refers to a causal antecedent question invoking a search through the causal chain representation of the instantiated script describing the story. If this search cannot provide a suitable answer

to a causal antecedent question, then a search is made in a 'plan' structure to which the script is attached. Information in the plan structure may then supply an answer to a goal orientation question having the same question concept.

2.8.3 Enablement

e.g. How were you able to afford a new coat?

Such questions are enquiries about states or their causative acts which enable the act described in the question concept. Although Lehnert does not refer to the fact, enablement questions are thus a special subset of causal antecedent questions.

2.8.4 Causal Consequent

e.g. What is the outcome of a plutonium leak?

The question concept is an action; the answer is given by the state that is the result of that action.

2.8.5 Verification

e.g. Is titanium oxide poisonous?

Verification questions need a yes or no answer, or some statement of the answerer's confidence in the truth of the question concept.

2.8.6 Disjunction

e.g. Who painted 'Beata Beatrix': Rossetti, Burne-Jones
or Morris?

The OR in disjunctive questions is not the logical OR (else a valid answer to the example question would be 'yes'); it is more accurately a collection of verification questions, one for each 'disjunctive' element, with only the true elements reported as answers.

2.8.7 Procedural/Instrumental

e.g. How did John get to London?

How do you make mayonnaise?

Questions in this category ask either about an object instrumental to the act in the question concept ('By train', in response to the first example question), or about a sequence of acts that together constitute the question concept act. Lehnert groups these two together, because they are very closely related. To see this, consider the answer 'by train' to the first example question. This is a shorthand for describing the collection of acts (in fact, a script) which constitute the procedure of travelling by train.

2.8.8 Concept Completion

e.g. Who cooked dinner last night?

The question concept for this category is an active conceptualisation. At least one slot or descriptive feature in the question concept is unknown (in the example, it is the identity of the ACTOR involved), and the question is answered by providing the appropriate feature.

2.8.9 **Expectational**

e.g. Why doesn't zinc rust?

If such questions were asked with a positive instead of negative force, they would be enablement or causal antecedent questions. (i.e. Why does iron rust?) The question concept is a negated conceptualisation indicating the violation of the enquirer's expectations. To answer such a question it is necessary to search for states in the causal chain which would disable the positive question concept or disable actions eventually resulting in the question concept.

2.8.10 **Judgmental**

e.g. What do you think of SDI?

Judgmental questions ask for a subjective opinion of the entity named in the question concept.

2.8.11 **Quantification**

e.g. How many cats do you have?

How ill is your uncle?

Questions in this category require counting (example 1) or a scale value (example 2).

2.8.12 Feature Specification

e.g. What colour is your car?

These are 'slot-filling' questions for stative conceptualisations, comparable to concept completion questions which 'fill slots' for acts.

2.8.13 Requests

e.g. Would you pass me that book?

Requests are instructions for action which are phrased as questions. They do not really expect a verbal reply.

2.9 Primitives and Representations for Domain Entities

It may appear that in CD theory, Schank has established a sufficient set of eleven primitive acts. These however are domain dependent. In [Schank & Carbonell 1979], the authors investigate the representation of the Gettysburg Address using the ideas of Conceptual Dependency. In exploring this, they make the point that the actions involved are not simple physical ones, involving individuals and everyday tasks. The eleven CD primitive

ACTs do not apply: they cannot be used as atoms in the construction of more complex molecules of meaning. Schank and Carbonell propose seven basic 'social ACTs' for use in representing social and political events. In considering the scope of the two available sets of primitives, they say:

"It is very unlikely that either the CD-primitives or the basic social acts would be useful in codifying, for instance, the knowledge relevant to understanding chemistry or microbiology. These domains require their own basic knowledge organizing units."

In the PROMPT system [Addanki & Davis 1985] described in section 2.5, processes are 'precanned', i.e. not described at a microscopic level, nor in terms of any primitives. For example, no relationship is evident between 'boiling' and 'freezing'. The description of processes includes specification of preconditions, the entities involved, the effects generated, and relevant mathematical equations. Weld [Weld 1985] is also concerned with the description of processes for use in models. He distinguishes between continuous and discrete processes. If the world state is described in terms of quantities which are either 'linear' (i.e. dimensional, having a continuous range of values) or 'nominal' (i.e. featural, having a discrete set of possible values), the nominal attributes cannot be affected by continuous processes, since there is no way monotonically to change a quantity 'with no inherent order'. Nominals are therefore affected only by discrete processes. Like

processes in PROMPT, Weld's processes are described in terms of their preconditions and influences.

2.10 Conclusion

Explanation in problem-solving systems is not an area where a vast amount of work has already been done. The achievements are modest, but the problem itself is a very difficult one. Sense can only be made of the field by regarding 'explanation' as a generic term, and attempting to discover what some of the specialised processes are that make up the rich fabric of the ways in which human beings explain what they know.

3 THE SCRIPT APPROACH TO CAUSAL MODELLING

3.1 Overview

The domain of the ADEPTUS system is the use of coatings applied to steel substrates to prevent or retard corrosion. The physical processes in the domain will be viewed as elements of an archetypal story or 'script' in Schank's terminology.

This chapter discusses the domain under consideration. Analysis of the knowledge obtained from the domain experts reveals how a causal model can be used as a representational formalism for domain processes. Following from this, Chapter 4 illustrates how the same formalism can be applied to the problem of selection within the domain.

David Hume described the origins of our ideas of causality thus:

"The necessary connection betwixt causes and effects is the foundation of our inference from one to the other. The foundation of our inference is the transition arising from the accustomed union. These are, therefore, the same." [Hume 1738]

This resembles Schank's comments about a script-based theory of understanding:

"In order to understand the actions that are going on in a given situations, a person must have been in that situation before. ...The actions of others make sense only insofar as they are part of a stored pattern of actions that have been previously

experienced." [Schank & Abelson 1977]

The "stored patterns of actions" refer of course to Schank's 'script' entities. Schank and Abelson are talking about social situations, and Hume is concerned with general causality. It seems reasonable to employ this 'causality as script' view in a system where perception of an external reality is not at issue.

3.2 The Outline Account

Analysis of the knowledge elicited from the domain experts showed that the physical processes involved in the use of anti-corrosive coatings form an archetypal story. The next few subsections show the essential elements of this outline account.

3.2.1 Substrate Preparation

First, the surface to be protected must be prepared. This is normally achieved by abrasion or impact methods; the substrate may be scoured by hand with wire wool or shotblasted by machine. The choice of method depends on several factors:

- (i) The size and shape of the substrate. Small intricate items cannot be successfully shotblasted. At the other extreme, large expanses

of flat or simply-shaped surface are not efficiently treated by hand.

- (ii) The location and mobility of the substrate. Shotblasting may be impractical in certain interior locations. However, if the substrate can be moved to a suitable place, the mechanical method may still be feasible.
- (iii) The availability of tools and personnel.

The aim of these mechanical processes is to clean the substrate and to provide a key, or degree of roughness, on the surface to be coated. The nature and depth of this key, and the extent to which impurities like rust or old coatings have been successfully removed, have a major influence on the subsequent adhesion between substrate and applied coating.

The substrate may also be degreased: that is chemically treated to remove existing adhesions of undesired substances, which create a thin but effective barrier between substrate and coating.

3.2.2 Coating Application

These preparatory activities should eventually lead to a prepared surface that is dry, clean and has a good key. A coating may then be applied to the surface. This is

achieved in one of several possible ways: these include brushing, rolling, spraying and tumbling. The characteristics of the individual paint will determine which methods are possible. For example, the high viscosity of bitumen coatings renders spray application completely impractical. Another major factor influences the choice of application method: the availability of skilled personnel. If the optimal application method for a coating is an airless spray, the method will nevertheless yield unsatisfactory results if employed by personnel untrained in the use of such equipment. In a case like this, one may well have to resort to brush application.

Possession of appropriate tools is also a necessary prerequisite to coating application. Eventually, using one or other of the possible application methods, adhesion between substrate and coating is achieved over the entire surface of the substrate. This means that on the substrate there exists a wet film, which will have a collection of measurable dimensions and observable properties.

3.2.3 Drying and Curing

The wet film is now left to harden. This may occur

(i) through drying by solvent evaporation

(ii) through curing or chemical crosslinking

or (iii) through a combination of solvent evaporation and

curing.

The type of hardening process is governed by the nature of the coating, but variables such as rate of hardening are affected by external factors including temperature, humidity and ventilation.

If the hardening processes include drying by solvent evaporation, a vapour will be produced. The chemical constituents of such a vapour are determined by the composition of the wet film, and it may be necessary to take account of potential vapour toxicity.

On completion of the drying process, a fresh dry film is obtained, adhering to the substrate. This film is described by properties and dimensions similar to the wet film, but the values along these dimensions have changed. For example, the film thickness is likely to have been considerably reduced, and the adhesion with the substrate enormously increased. Some of the features of the wet film that were descriptive of the liquid state will have been replaced by new descriptors appropriate to a solid substance. For example, the concept of viscosity is relevant to consideration of the wet film; in the dry film this has no meaning and can be replaced by such qualities as hardness, flexibility etc.

At this point in the history , in practical terms, the coating begins to serve its purpose.

3.2.4 Wearing

Once in service, the dry film can be affected by many factors. It is convenient to group these into environmental factors and sub-film factors. The environmental factors are any which are external to the film-substrate system, and include:

- (i) Chemical contact: acids, alkalis, organic compounds etc.
- (ii) Water contact: fresh water, salt water, permanent immersion, splash zones.
- (iii) Mechanical damage to the film: impact, abrasion, stressing.
- (iv) Radiation: e.g. oxidation effects causing yellowing of white pigments.
- (v) Atmospheric pollution: industrial contaminants, ozone.
- (vi) Temperature: high temperatures, low temperatures.

Sub-film factors are those arise from processes occurring either within the film or at the film-substrate interface. For example, if the substrate was wet when the coating was applied, pockets of moisture may be trapped. An increase in temperature can then vapourise this moisture, causing 'blisters' to form.

All such factors take their toll on the dry film. As time passes, the attributes of the film will change in response

to these destructive processes.

3.3 The Script

When the preliminary knowledge analysis was complete, it became clear that the sequence of processes that constitute the foregoing description can be represented using a modified version of Schank's scripts. It is this idea that underpins this thesis.

Figure 1 illustrates an initial high-level representation of such a script.

3.4 Active Images

Each of the ACTs can be thought of as a reference to an entire sub-script rather than a primitive ACT. For example, the surface preparation is not a simple ACT, but a term used as a shorthand description of a complex, variable set of ACTs. This is analogous to Schank's idea of a MAINCONS, the ACT central to the script, that is sufficient to use if the entire script is not needed.

Although causal chaining is used following Schank (see section 3.6), it is not appropriate to use his Conceptual Dependency primitives for this domain. CD theory requires

the use of primitives to obtain a language-independent representation of meaning, in which the essential ideas common to whole classes of verbs can be made explicit. While Schank's eleven primitive ACTs are sufficient for

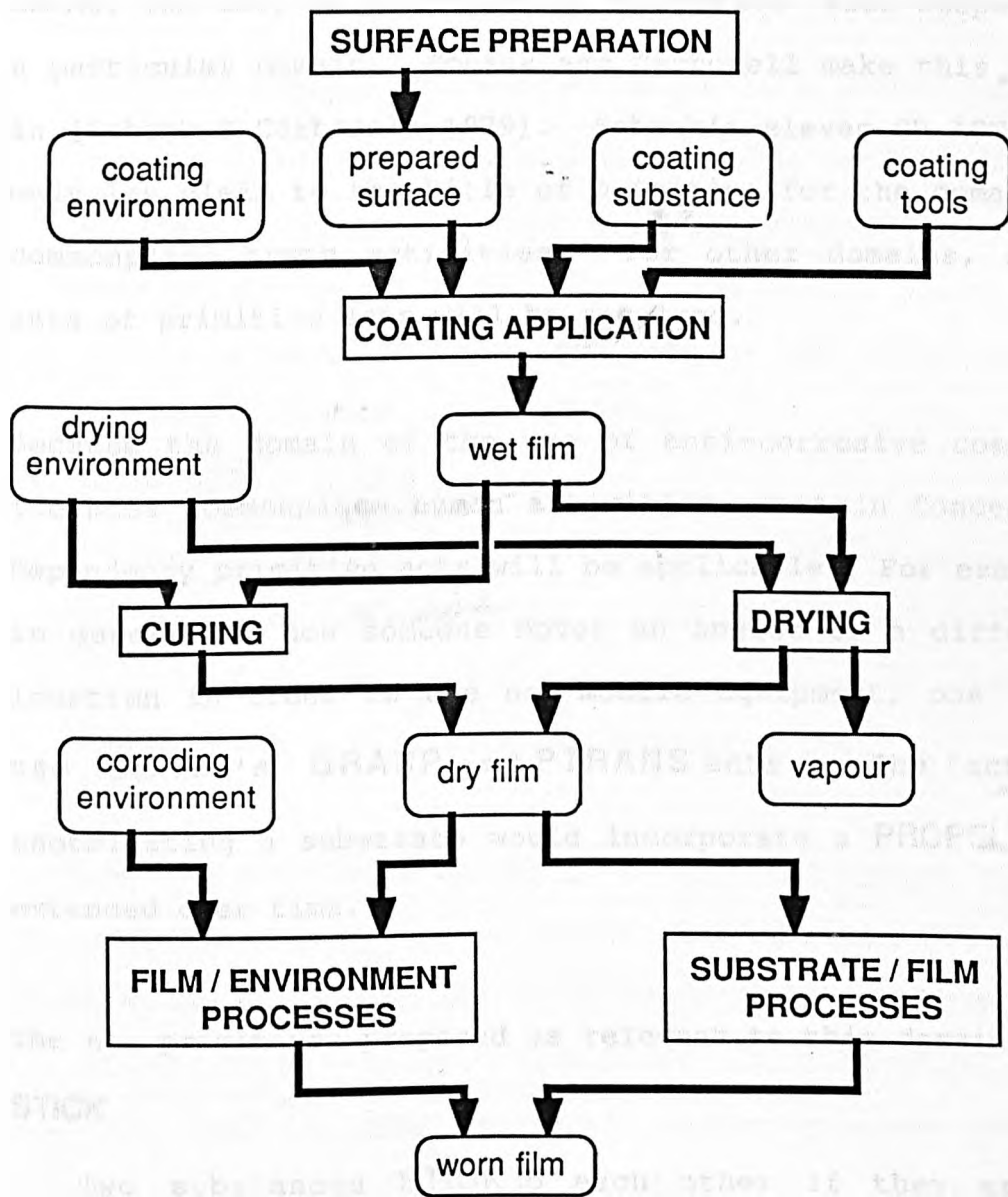


Figure 1

the representation of all the active images in his chosen domain, they do not offer a vocabulary with which to describe physical processes largely independent of human motivation.

An ACT can only be described as 'primitive' with respect to a particular domain. Schank and Carbonell make this point in [Schank & Carbonell 1979]. Schank's eleven CD ACTs can only lay claim to the title of primitive for the domain of commonplace human activities. For other domains, other sets of primitive acts will be required.

Because the domain of the use of anti-corrosive coatings includes commonplace human activities, certain Conceptual Dependency primitive acts will be applicable. For example, in describing how someone moves an object to a different location in order to use non-mobile equipment, one could use Schank's GRASP and PTRANS acts. The act of shotblasting a substrate would incorporate a PROPEL act, extended over time.

The new primitives proposed as relevant to this domain are:

STICK

Two substances **STICK** to each other if they are in physical contact, and remain so over a period of time.

STATECHANGE

A **STATECHANGE** process is recognisable in that the

major participant is present both before and after the process but in different physical states, e.g. solid, vapour etc. Thus STATECHANGE is at the root of processes such as evaporation, condensation, freezing, melting, and sublimation.

TOUCH

Physical contact of two substances is a state which is brought about by a TOUCH process. The process implies movement of one substance with respect to the other. This primitive can be used to represent the concepts of impact (hence shotblasting, spray application and mechanical damage) and spillages.

CHEMCHANGE

In such a process, one or more participating substances undergo chemical reactions to produce a chemically new participant substance. Examples are curing (chemical crosslinking), oxidation and precipitation.

In order to use the CD theory in a technical domain such as this, I have found it necessary to make an important distinction between an ACT and a PROCESS. An ACT must have an ACTOR: some entity (sentient or otherwise, e.g. gravity) that performs or is the causative agent for the occurrence. A PROCESS, in contrast, has participating entities, but none of these can be considered the causative agent for the occurrence. If this distinction were not made, then when representing, for example, the adhesion

between paint and substrate, one would be forced to ask whether the paint was the ACTOR of the STICKing act, or whether it was rather the substrate. The introduction of the idea of PROCESS moves away from Schank's strong focus on the ACTOR-ACT combination as the centre of interest.

This distinction between ACT and PROCESS avoids forcing the representation of domain-level active images uniformly into Schank's

ACTOR ACTION OBJECT DIRECTION (INSTRUMENT)
CD syntax. For PROCESSES, a simple representation can be employed, comprising the name of the PROCESS, followed by the participants in that process. For example:

STICK coating substrate

STATECHANGE coating

So, in general,

PROCESS <participant₁> <participant₂>... <participant_n>

3.5 Stative Images

3.5.1 Schank's Stative Images and the Cognitive Psychology

View

Schank's representation for states in Conceptual Dependency Theory is simple and generated ad hoc. A stative image

takes the form

Object Attribute

or

Object Attribute Value

For example,

John ANGRY

Can OPEN

John HEALTH (-10)

Schank's scheme fits well with a simplified approach to the classical view of concept definition in cognitive psychology. The classical view describes concepts in terms of 'features' and 'dimensions'.

A feature is a qualitative attribute, having a binary present/not-present nature. This accords with examples like

John ANGRY

Can OPEN

A dimension is a quantitative attribute; that is, one which can be used in magnitude comparisons with the same attribute of a different concept. For example, if a Sabatier knife is very sharp and a table knife is quite sharp, then the Sabatier knife has a greater magnitude in the sharpness dimension than does the table knife. This of course maps very well on to Schank's use of scales; the value between -10 and +10 is the magnitude of the

dimension.

3.5.2 Applicability to the Anti-Corrosive Coating Domain

To a very great extent, the stative images in Schank's work are of only secondary interest. His story domains are highly action-centred; states of the world are only incidentally important insofar as they enable or are the reason for the next action. His decision to use an episodic representation for information encompasses an exclusion of semantic representations; Schank admits that this may not be appropriate for more scientific domains [Schank 1975]. Lehnert was dissatisfied with the Conceptual Dependency representation of everyday physical objects, and proposed a primitive functional description of items such as taps, cups and plates for use with the script-based formalism.

The items to be described in the anti-corrosive coating domain, however, must be given a richer and more detailed representation. It is, after all, the coating film and its relation to the substrate that are of prime importance.

The representation of stative images in ADEPTUS is clearly divided into two separate levels, the gross and the detailed. At the gross level the representation of the world state is a very simple featural one. For example, the

state immediately preceding the STICK process may have the form

PHYSCONT coating substrate

borrowing the PHYSCONT (physical contact) relation from Conceptual Dependency. The gross level delineates only the names of the entities which are relevant to the portion of the archetypal story into which the state fits, coating and substrate in the example. This may be supplemented by simple coarse-grain relationships between the participating entities i.e. PHYSCONT. The coating and substrate are here features of the 'world state' concept.

The representation of the more detailed attributes of the world state takes the form of a collection of features and dimensions belonging to each of the gross world state entities, e.g. coating. It is possible to predict the names of the attributes that state-descriptive entities can possess. For example, any film will have a thickness, a colour and a physical state (solid or liquid). Some of these attributes are features in that the possible values of the attributes form a discrete set, e.g. colour. Others, like film thickness, are clearly dimensions.

It is the changes in the values of attributes at this detailed level that are of major importance to problem-solving in the domain. At the gross level of state description, information is held only on whether the

coating film and substrate remain in general physical contact. At the more fine-grained level, the details of the changes in the environment/film/substrate system can be viewed.

The feature description of the gross world-state is concerned only with the existence or otherwise of entities. A variety of attribute names is not required to link the world-state concept and its attributable entities. Figure 2 illustrates an instance of the world state: note that all the arcs are simply labelled 'has-feature'.

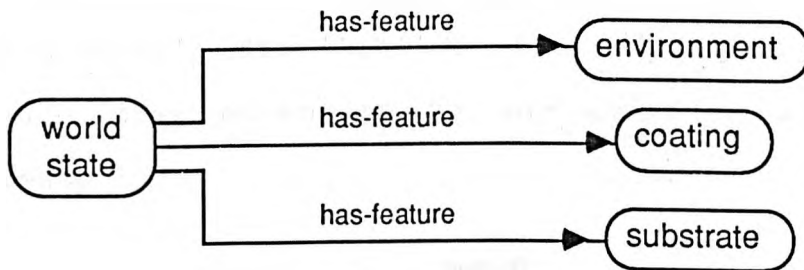


Figure 2

The predictable set of attributes for individual entities on the detailed level produces a proliferation of arc names. Figure 3 shows an example of this. For these reasons, the abstract representations of the two levels have been translated into different formalisms. The gross level has a representation in list form, where the elements of the list are attributes of the world state. The

detailed level is cast into frames. This permits prior, hierarchical determination of appropriate attribute names, and the facility to restrict the values that are considered valid for a given attribute.

3.6 Causal Chaining

Categorising active images into those involving ACTs and those involving PROCESSES forces a reexamination of the principles of causal chaining. The continuing applicability of the causal syntax in the presence of PROCESSES must be questioned. The substitution of 'Acts or Processes' for 'Acts' in the rules of the causal syntax will draw attention to any conflict with commonsense reasoning.

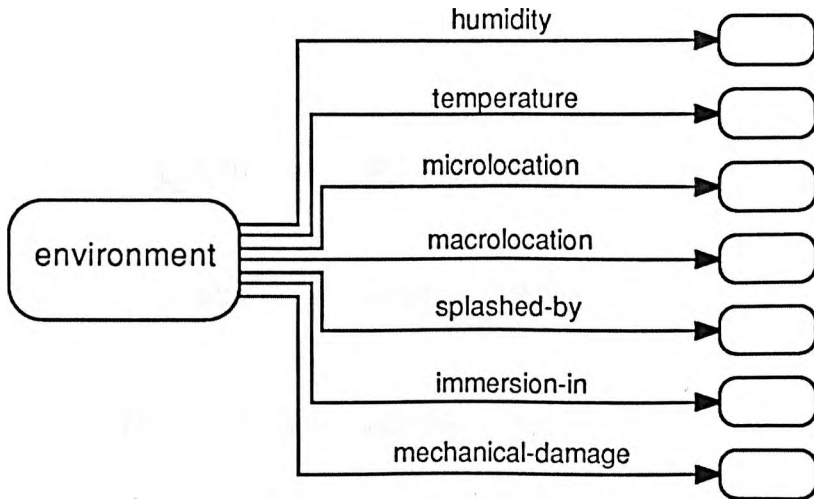


Figure 3

1 Acts or processes can result in state changes.

ACT => STATE

r

PROCESS => STATE

r

2 States can enable acts or processes

STATE => ACT

E

STATE => PROCESS

E

3 States can disable acts or processes

STATE => ACT

dE

STATE => PROCESS

dE

4 States, acts or processes can initiate mental states

STATE => MENTAL STATE

I

ACT => MENTAL STATE

I

PROCESS => MENTAL STATE

I

5 Mental states can be reasons for acts or processes

MENTAL STATE => ACT

R

MENTAL STATE => PROCESS

R

Rule 5 clearly does not make sense when 'process' is substituted for 'act'. A mental state can only be the reason for an act when the mental state is attributable to the actor of the ensuing act. Since a process has no actor, a mental state cannot be a REASON for that process. It is therefore necessary to restrict Rule 5 to the version given by Schank:

5 Mental states can be reasons for acts

MENTAL STATE => ACT

R

However, considering the nature of the domain of interest in ADEPTUS, it seems possible to dispense with the use of causal syntax rules 4 and 5 altogether. The extent to which the system will be concerned with people's mental state is minimal. Doubtless a case could be made for the representation of thoughts or knowledge in the minds of the process operatives required by the preparation or application phases, but the relevance is peripheral. I shall, therefore, be content to describe such operatives as 'skilled' or 'unskilled' rather than attempt to detail the minutiae of their mental states.

This constitutes in effect a restriction on the domain. A

precedent exists in Schank's own work; in stories about spilling beer, he remains interested only in people's actions and reactions. He makes no attempt to represent or deal with concepts like liquid soaking into fabric or the chemical changes bringing about staining.

3.7 Conclusion

Knowledge analysis suggested the central theme of scripts as a representation for physical processes. However, Conceptual Dependency could not offer a range of primitives appropriate to the anti-corrosive coating domain. In examining the kinds of verbs used in discussing the domain, the distinction between ACTs and PROCESSES became evident, and this was incorporated into the selection of new primitives.

For the ADEPTUS system, the causal syntax has been restricted to rules 1 - 3, which are all those that do not mention mental states. The stative images that are of central interest in this domain have two levels of representation. The uses of these two levels will be explained in Chapter 5.

4.1 Overview

Having chosen to represent the physical processes of the domain of interest in a script form, I now turn my attention to the problem-solving aspect of the desired system. The task of interest is the selection of the optimal coating from a specified set in given circumstances. However, the essence of describing the causal model separately from the problem-solver is that problem-solving strategies for other tasks in the domain may be described and used without altering the causal model. Therefore the task of coating selection will be described in terms of a manipulation of the causal script. I shall seek to couch this description in terms of a small set of possible manipulations of a causal script, in the hope that this set will also then provide the building blocks for problem-solving descriptions for other tasks.

The problem-solving process is represented in ADEPTUS by a script entity. This outlines the essential acts which ADEPTUS must perform on an object script which is a representation of a causal model of the domain of interest. Each act in the problem-solving script can be located within a hierarchy of script-manipulative tasks. This hierarchy is differentiated on the basis of the object

type(s) accessed and the essential direction of search. Following Lehnert's approach to question-answering using scripts, questions can be answered by a search for a match for the question concept, followed by one or more script-manipulative tasks. The nature of the task is dependent upon the question category.

4.2 Paint Selection from Industrial Guidelines

At the heart of the industrial guidelines for paint selection is a matrix of environmental descriptions against paint class. A simplified version is illustrated in Figure 4.

The user can extract from this matrix the rows which apply to the particular situation under consideration. Examples are given in Figures 5, 6 and 7. This row extraction exercise leads to a preliminary selection of one or more paint classes by determining those which are not indicated as inappropriate for any of the conditions listed. The user is then referred to paint class information sheets, where more detailed data on coatings can be found. The information sheets typically encompass such information as application methods, toxic components, drying characteristics, single film thickness, working temperature ranges, pH tolerance limits, colour availability and cost. The user must examine this information to see which set of





































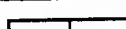



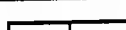














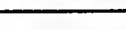
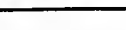






















KEY	Very suitable 	Acceptable 	Inappropriate 	PAIN T C L A S S	PAIN T C L A S S	PAIN T C L A S S	PAIN T C L A S S
				A	B	C	D
Surface Preparation Sa3, Sa 2.5							
Surface Preparation Sa2, St3							
Surface Preparation St2							
Indoor heated							
Indoor unheated							
Exterior sheltered							
Exterior exposed, inland/unpolluted							
Exterior exposed, coastal/industrial							
Salt water immersion							
Fresh water immersion							
Weak acid spills							
Strong acid spills							
Weak alkali spills							
Strong alkali spills							
Organic solvent spills							
Product contamination							
Mechanical impact/abrasion							
High temperatures (>80 C)							
Low temperatures (<0 C)							

Figure 4

<u>Tank in Soap</u> <u>Manufacturing Industry</u>	PAINT CLASS A	PAINT CLASS B	PAINT CLASS C	PAINT CLASS D
Surface Preparation Sa2				
Indoor Unheated				
Weak alkali spills				
Organic Solvents				
Mechanical Impact / Abrasion				

Figure 5

KEY
 Very suitable
 Acceptable
 Inappropriate

<u>Ship Hull</u>	PAINT CLASS A	PAINT CLASS B	PAINT CLASS C	PAINT CLASS D
Surface Preparation Sa3, Sa2.5				
Exterior exposed, coastal / industrial				
Salt water immersion				
Salt water splash zone				
Mechanical Impact / Abrasion				

Figure 6

paint characteristics most nearly fits the constraints of the situation.

Problem descriptions coded in matrices like those described raise some interesting points:

(i) A solution is not always available. In the examples given, this is largely a result of the reduction in the number of paint classes considered. Clearly, coatings that are suitable for ship hulls must be available.

<u>Suburban Garden</u> <u>Fencing</u>	PAINT CLASS A	PAINT CLASS B	PAINT CLASS C	PAINT CLASS D
Surface Preparation S2				
Exterior exposed, inland / unpolluted				
Low temperatures (<0 C)				
Mechanical Impact / Abrasion				

Figure 7

(ii) In situations where no completely satisfactory solution exists, a choice must be made among the more or less unsatisfactory solutions. The aim is to find the best coating, not necessarily a perfect (or even very good) one. This necessitates an examination of the relative importance of those constraints that force a paint to be deemed unsuitable.

For example, consider the selection of a coating

for a garden fence described in Figure 7. Of the four given constraints there are three that rule out exactly one coating each; i.e. surface preparation to St2 standard (a high quality hand-prepared surface) rules out coating A, mechanical impact rules out coating D and low ambient temperatures rule out coating B. (Coating C is ruled out on two counts, so it seems reasonable to view this as being more inappropriate than the other three). If further examination of the constraints shows that a surface preparation of St2 standard cannot be improved upon, that winter temperatures can be relied upon to produce frequent and prolonged icy environments, but that impact is only a fairly infrequent occurrence (caused, for example, by a child's bicycle), then clearly coating D is preferred over A and B.

- (iii) The gradient of response to a constraint is not easily perceptible. In the soap manufacturing process tank example, if it were possible to prepare the surface more thoroughly to standard Sa2.5 (a medium-quality shot-blasting), coating A would show optimal performance for all the given constraints, thereby superceding the initial choice of coating B. On reference back to Figure 4, it can be seen that this is caused

by a sharp drop in suitability as the preparation standard drops from Sa2.5 to Sa2.

4.3 Strategies for Selection

The solution space for the task of selecting the optimal coating can be partitioned hierarchically. That is, there are about a dozen broad classes of coating, many of which can be divided into subclasses. Specific commercial coatings form the leaves of such a tree.

In the original design for ADEPTUS, it was intended that paint selection would be performed in stages, conforming to the coating hierarchy. A first pass would use knowledge about each of the major classes, and would instantiate a script for each, eventually producing the optimal paint class. This process was then to be repeated (with the user's agreement) using knowledge of subclasses of the selected class; and so on until a specific paint was reached.

From an industrial perspective, this scheme would be most inefficient. A full script-based system used in this way for problem-solving would require vast resources, and enormous amounts of information produced inside the system (i.e. the details of all the instantiated scripts) would never be accessed.

The second design for ADEPTUS took these criticisms into account. Scripts are used to lend explanatory power to the problem-solving system. However, those scripts are no longer intended to deal with the entire process of selection. The current scheme is to use a simple rule-based system to identify perhaps three or four paint classes that cannot be ruled out in a very simple fashion. This process corresponds to the selection matrix in Unilever's text guidelines. Having established this shortlist, the script-based system can be used to predict the performance of each candidate solution, and to compare and rank those performances.

If a simple system based on the propagation of a certainty-factor through a rule network (a PROSPECTOR-style system) is used to make a preliminary selection, the kind of results obtained may be

	Final Probability
Paint class 1	0.84
Paint class 4	0.81
Paint class 6	0.79
Paint class 7	0.53
Paint class 2	0.42
Paint class 3	0.21
Paint class 9	0.19
Paint class 5	0.19
Paint class 8	0.11

The major drawback in terms of explanation in such a system lies in the inability to decode the subtleties of meaning implicit in the single number used to represent the suitability of a coating. The immediate questions posed by results in this form are:

Are the differences between the results for paint classes 1, 4 and 6 significant ?

If so, what do they imply ?

The task of ADEPTUS is to help the user to understand the implications of the use of these competing coatings in the problem situation. Running the script-based system will give a judgment on each coating in terms of values for many meaningful characteristics of the eventual worn paint film. This is itself an explanation of some of the results of the first coarse selection phase.

After the first phase, the user may wish to examine detailed predictions for coatings that seem less promising. This may occur, for example, if a large stock of a particular coating is already available, but the coating is not initially recommended. The methodology of ADEPTUS allows the user the opportunity to explore courses of action that would otherwise be too risky to try.

Both of these uses of the script-based system only require it to make predictions of the eventual outcome, and in conventional terms these are the 'problem-solving' uses of the system. The general question-answering facilities

afforded by such a formalism are discussed in Chapter 6.

4.4 A Script for the Selection Task

The industrial guidelines provide an indication of the portions of the causal model that must be instantiated to describe the situation constraints. Figure 8 shows the information aspects of the 'story', which must be provided by the user, indicated by the shaded area.

Armed with such information, it is now possible to use the causal model by supplying a description of a coating and the necessary tools, and 'running' the model. The coating application acts and processes take place, instantiating as they do so a description of the resulting wet film. The drying and/or curing processes then play their part, modified by the known environment during the transition from wet film to dry. Having established the expected nature of the new dry film, the effects of the corroding environment can then be judged, giving a view of the dry film after wear.

This running of the model produces an instantiated script providing a great deal of information about a particular coating. Selection of the optimal coating must take place by a comparison of the resultant worn film states of each of the instantiated scripts. This comparison takes place

in two distinct phases. First, the worn film description can be compared with an 'ideal' worn film; that is to say,

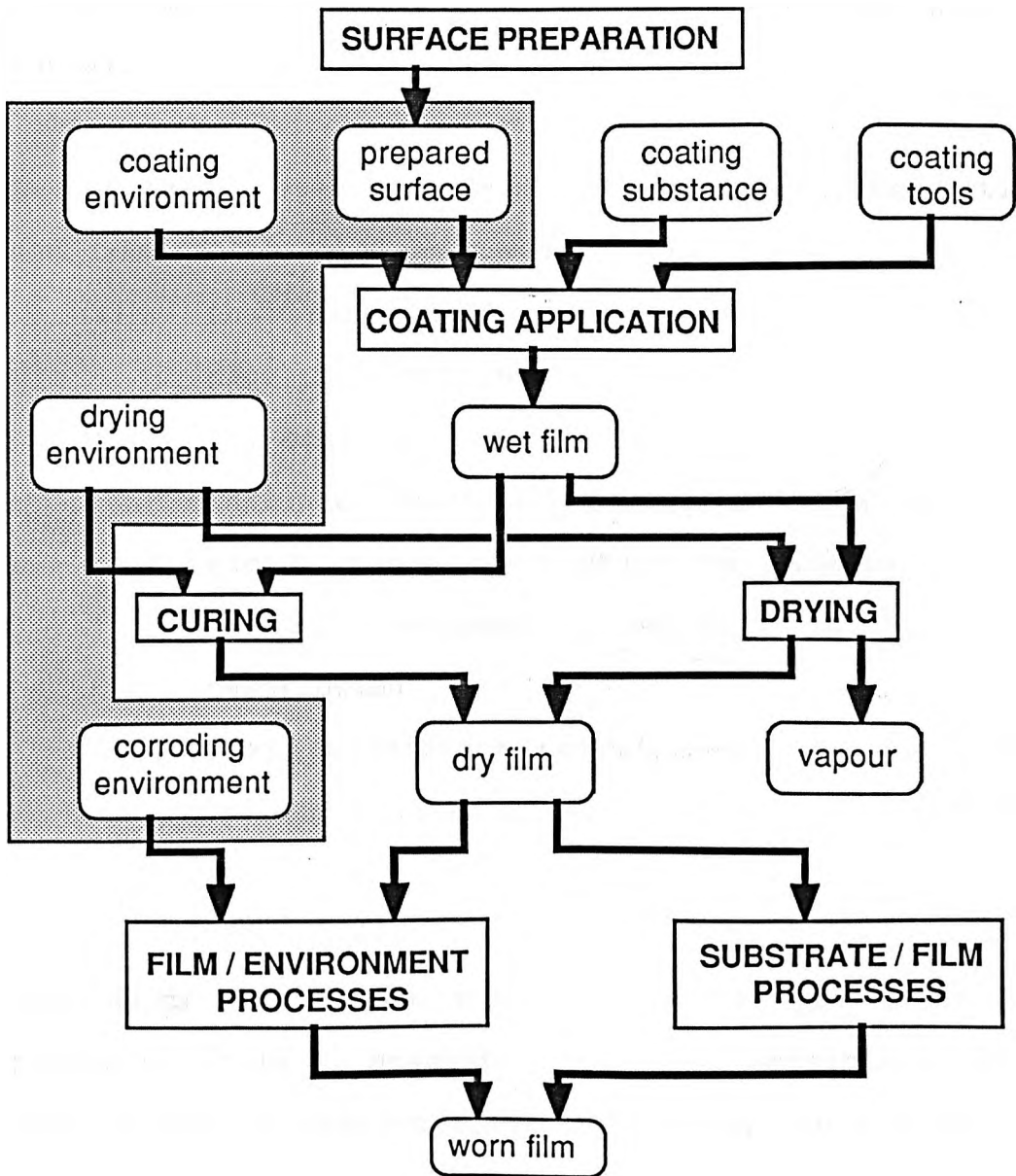


Figure 8

one which has optimal film/substrate adhesion, uniformity of thickness etc. It is these divergences from the ideal

that will then be compared to establish some ranking of the coatings. No judgments of appropriateness are generated at the domain level; such judgments are made as part of the problem-solving task, not at the level of physical causality.

Let us take a step back from this problem-solving activity and examine the outline of what is taking place.

- (i) the domain level script is instantiated with the problem constraints.
- (ii) for each solution of interest, the causal model is run to obtain a description of the outcome.
- (iii) each outcome description is compared with an 'ideal' outcome, producing a divergence description.
- (iv) these divergence descriptions are compared to obtain a ranking of the associated potential solutions.

Just as in chapter 3 a script for archetypal domain level processes could be described, so it is possible to detect here a problem-solving script outlining the essence of a selection strategy. This script outline is quite independent of the domain of interest: it is equally applicable to antibiotic therapy selection or to the selection of standard steel formulations for oil rig pipes. A clear constraint, however, is that the available solutions must form a discrete set, or at least that

solution spaces having continuous-variable attributes can be 'sampled' in such a way as to yield a meaningful collection of discrete solutions.

Representation of physical processes and problem-solving strategies within the same abstract formalism has the appeal of elegance, but the representation of problem-solving must be examined more closely. It is clear from the summarised description above that the causal chaining principle can be used to connect together script-manipulative acts and the resulting states of knowledge (Figure 9). It is now necessary to examine the entities filling the places of active and stative images.

4.5 Stative Images

There are two major categories of stative image in the problem-solving script. The first is the DOMAIN-LEVEL (d-level) image, which is either a d-level script or a recognisable part of one. Examples are an environment description, a description of a gross physical state (e.g. a coating film in physical contact with the substrate), or description of a potential solution. The problem-solving script (the SELECTION-LEVEL or s-level script) is acting upon or manipulating a d-level script to fulfill its goal of becoming as fully instantiated as possible, just as Schank's ACTORS act on objects in their environment to

achieve their goals.

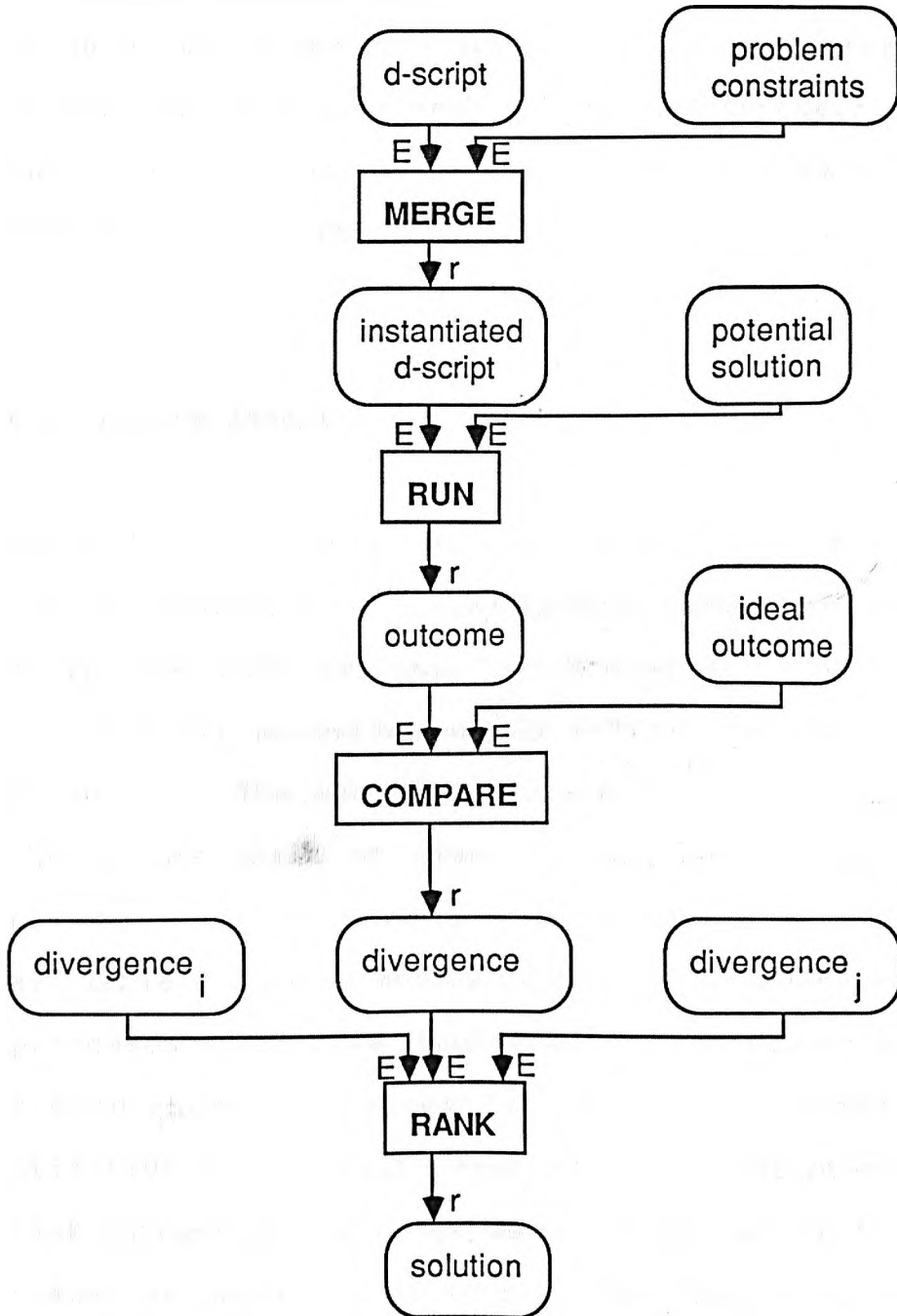


Figure 9

The second major category comprises images that are

abstractions from the d-level. This encompasses the divergence descriptions and the solution ranking list. These abstractions must of course contain references back to entities in the d-level script used in their generation, but do not contain the quantity of knowledge carried in that d-level script.

4.6 Active Images

The ACTs or PROCESSEs defined for use in an s-level script can be approached by identifying classes of tasks which manipulate other scripts. In Chapter 2, Lehnert's approach to question answering using script representations was discussed. The act of answering a question requires that the causal chain or image of interest be searched, and possibly used for inference if the answer is not explicitly available and hence susceptible to simple retrieval. These processes thus have identical characteristics to those needed here. Lehnert's categories (like Schank's primitives) are established ad hoc. Script-manipulative task categories can be established and Lehnert's categories viewed as instances of these. See Chapter 6, section 6.2 for a detailed discussion.

A TASK is a manipulation or examination of a script-entity, or part of such an entity. Instances of tasks are the elements of the problem-solving s-level scripts, and can be

described as starting with evidence and searching for a conclusion or solution to a 'primitive problem'. The evidence or problem description refers to the state of the script under scrutiny before the task commences; the solution or conclusion is comprised of the script characteristics that are present after completion of the task. Viewing the d-script as a causal chain, it is possible to categorise the possible manipulations, thus describing valid task types.

4.7 Classification of Script-Manipulative Tasks

4.7.1 The Causal/Acausal Differentiation

The first major division proposed is:

(i) CAUSAL TASKS

A class of tasks in which the relationship between the evidence or problem description and the set of possible solutions involves a representation (whether implicit or explicit) of physical causality. Such tasks are susceptible to domain-level representation as causal chains.

(ii) ACAUSAL TASKS

A class of tasks in which the problem description and problem solution have only a non-causal relationship. Such tasks are not susceptible to domain-level representation as causal chains, but involve only individual images or fragments of images, i.e. individual entities.

4.7.2 Differentiation of Causal Tasks

4.7.2.1 Basis of the Differentiation

The class of causal tasks can be further decomposed on the basis of the temporal relationship between the point in time at which the problem is posed, and the physical events which are symbolically represented.

4.7.2.2 Analytic Tasks

A causal task may be ANALYTIC. The subclass is categorised by an attempt to discover what past events or world states could have led to the observations made. Using the causal chaining representation, let us examine a sequence describing the reaction between water and sodium, illustrated in Figure 9.

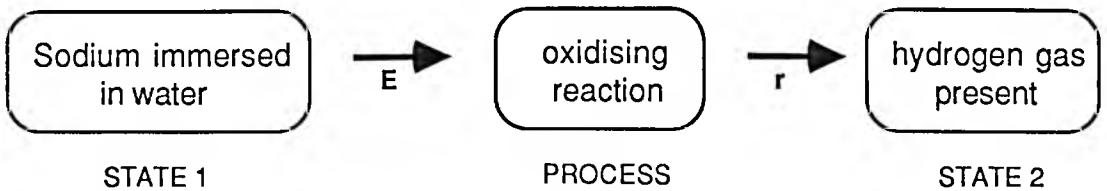


Figure 9

An analytic task seeks the CAUSE of the state observed; in this case the analytic task would be seeking the cause of the presence of the hydrogen gas. The time at which the problem is posed is when State2 is in existence, and the solution to the problem is obtained by working back down

the causal chain. In general, then, an analytic task has a causal chain representation like that shown in Figure 10.

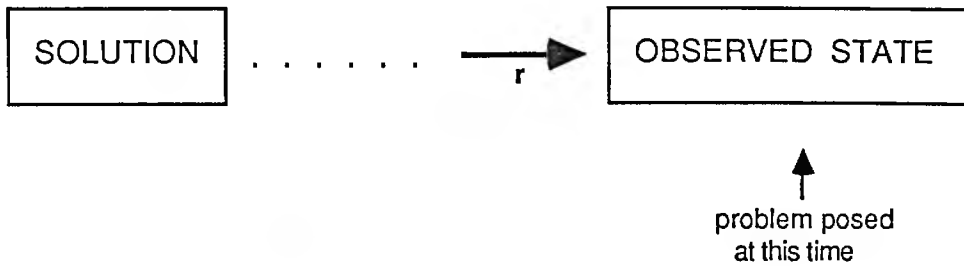


Figure 10

The SOLUTION here is not implied to be the act syntactically required to fulfill rule 1 of the Causal Syntax; it is merely indicated to be somewhere in this portion of the causal chain.

4.7.2.3 Predictive Tasks

The second fundamental category of causal task is that of the PREDICTIVE task. Tasks in this subclass look to the future rather than the past, and attempt to predict an outcome of a domain situation described to the problem-solver. Using the example of the oxidation of sodium, a predictive task would be to describe the outcome of the situation in which a piece of sodium is immersed in water. The predictive task represented as a causal chain therefore has the general structure shown below.

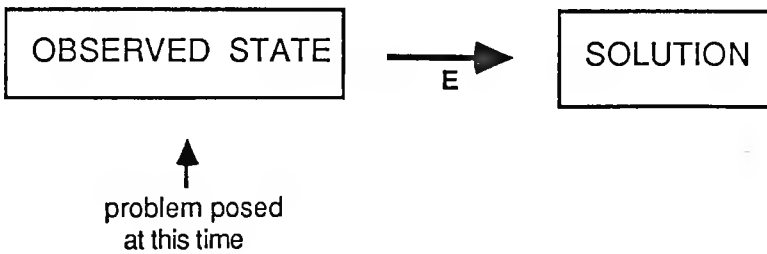


Figure 11

The observed state is that sodium is immersed in water. This state enables certain processes, and it is a description of these processes that constitute a solution to the predictive task.

4.7.3 Differentiation of Acausal Tasks

4.7.3.1 Basis of the Differentiation

Acausal tasks can be described as RELATION-SEEKING. They have no causal chain representation; they are tasks which take as their starting point limited descriptions of one or more world states, entities, processes or acts, and attempt to identify currently unknown descriptors for those images or entities.

The class of acausal tasks can be further decomposed on the basis of the number of states or acts which are presented as the problem definition.

4.7.3.2 Intraconceptual Tasks

An acausal task may be INTRACONCEPTUAL, that is, concerned only with a single state or act. This subclass is therefore characterised by an attempt to discover additional, currently unknown, information concerning a unique state of or act within the domain. For example, we may describe an INTRACONCEPTUAL task concerning a state:

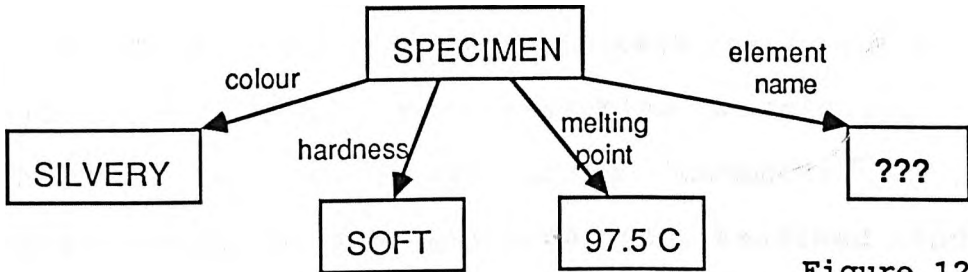


Figure 12

Or, concerning an act:

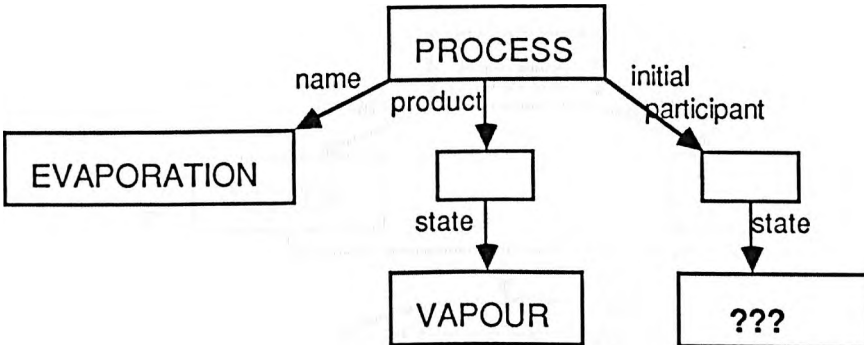


Figure 13

Intraconceptual tasks are 'slot fillers', where the slot being filled is a part of the description presented in the problem description.

4.7.3.3 Interconceptual Tasks

The other basic subclass of acausal tasks is that of INTERCONCEPTUAL tasks. Such tasks are characterised by their use of two or more distinct concepts in the problem definition. Interconceptual tasks are capable of using concepts from different levels: this leads us to define two separate subcategories.

An ABSTRACTING interconceptual task has two or more domain-level concepts as its problem description. Thus such tasks can be thought of as 'comparative' tasks. Similar concepts are provided and examined and the differences or similarities are sought; or the submitted concepts are ordered on the basis of some specified attribute. For example:

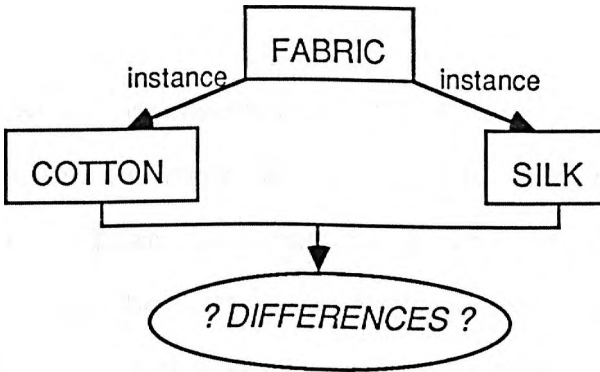


Figure 14

Here, the values of corresponding 'slots' for each item are involved; and the answer is not a slot value belonging to either item. For example, both cotton and silk may have

the property 'origin'. The value for silk will be animal, and the value for cotton will be vegetable. The solution to the abstracting interconceptual task certainly concerns the 'origin' property of the fabrics, but the 'value' attributable to some 'origin' slot in the solution will be a more complex structure than a value in a fabric schema. It is unlikely that the difference would be regarded as a 'property' of either or both items; rather is this difference only meaningful for the combination of items. The task has provided as its solution a state description which is unlike the states that formed the problem definition. The solution does not describe a real-world object, but instead is referring to a more abstract concept involved in problem-solving. The abstracting interconceptual task has moved from the domain-level states described in the problem definition to an s-level entity produced as its result.

A SPECIFYING interconceptual task has a heterogeneous set of state descriptions with respect to the level of the description. That is, one or more states in the problem definition must be at the d-level and (at least) one at the s-level. The fundamental idea here is to take something known about how a problem can be solved (the s-level concept) and some general knowledge about the domain of interest (a d-level concept). By combining these pieces of information from different levels, a more specific picture of a possible state of the domain can be

created. The detailed domain state can then be used in various causal tasks to obtain further domain-level information.

For example, the description of known solutions in ADEPTUS is an s-level concept. This can be merged with the d-script 'in parallel' to yield alternative worlds, each world detailing one of the possible solutions. This 'shattering' of the possible solutions is an example of a specifying interconceptual task.

4.8 Conclusion

A pattern can be detected in the actions required to make a selection from the known solution set of the anti-corrosive coating domain. This typical sequence of actions has, like the domain level processes, been represented in script form.

The s-level script is concerned with manipulations of the domain-level script and individual concepts. The possible manipulations have been described in a set of primitive tasks organised into a hierarchy. This task hierarchy is independent of the domain of the script which is being manipulated. The categorisation of the primitive tasks will be of central importance when Lehnert's question classification is reviewed in Chapter 6.

The next chapter concerns the details of the current implementation of the ADEPTUS system.

5 DESCRIPTION OF THE ADEPTUS SYSTEM ARCHITECTURE

5.1 Introduction

The ADEPTUS system is a knowledge base centred on the script formalism, employing demons, rulebases and procedural knowledge where appropriate. It is implemented in SRL from Carnegie-Mellon Intelligent Systems Laboratory. SRL was chosen as the only language available to me at that time which offered a powerful frame-based representation, supporting inheritance (both automatic and user-defined) and 'contexts' to allow independent reasoning in simultaneously existing 'parallel worlds'. The initials SRL stand for Schema Representation Language, and an enhanced version of the language is now commercially available under the name of KnowledgeCraft.

SRL is written in Franz Lisp. Its basic representational formalism is the **schema** or frame. A schema has several **slots**, each of which can hold one or more values. Often, the value of a slot is the name of another schema. Thus the slots act as links between schemata. When describing a schema in a diagram, I shall adopt several notation conventions: these will be explained when they are introduced.

Inheritance via *is-a* and *instance* links between schemata is

automatic; other inheritance paths and inheritance types can be defined by the programmer. As an example of a programmer-defined inheritance path, consider the following. If a schema used in a script describes a state, some values in that state may be altered as progress is made through the script. We might then describe the initial state as **schema1**, and the next alteration to this state as **schema2**. Links can then be established to assert

```
{ { schema2
  is-a classname
  successor-to schema1 }
```

Figure 15

This illustrates the schema **schema2**. The definition of the entire schema is surrounded by double braces. The first symbol inside the braces is the name of the schema to which the definition belongs. Below this is a collection of pairs, each pair consisting of a slot name (in italics) and the associated slot value. In the above example, there are two slots, *is-a* and *successor-to*. The value of each of these slots is a schema name. (Bold typeface is used for the names of schemata.)

If we define suitable inheritance characteristics (in the form of an inheritance schema) for the *successor-to* link, it becomes possible to access any slot and slot value of **schema1** from within **schema2**. Unlike the *is-a* link which

denotes the position in the class-subclass hierarchy, the *successor-to* link is not intended to denote that **schema2** is a subclass or element of a **schema1** set, but merely indicates that unless information is available to the contrary, we can assume that values attributed to **schema1** are still valid within **schema2**.

Several types of inheritance can be defined within SRL. For example, the definition of a mapping inheritance specification schema allows inheritance of a modified slot value from one schema to another. So if for instance the *thickness* slot of **schema1** has a value of 100, we could define a mapping inheritance specification schema in **schema2** for the *thickness* slot, such that the value of the **schema2** slot, if obtained by inheritance, was 10% of the unmapped inherited value. Thus, if we request a value for **schema2** *thickness* and the value can only be obtained by inheritance from **schema1**, then the inherited value will be returned as 10.

Within SRL, schemata can be defined within 'contexts'. These contexts can be arranged hierarchically, and any schema within a context is permitted to inherit slots and values from the same schema in an ancestor context. For example, a general version of the domain-level script **\$d-script** is established in a schema within the root context. (This is the SRL system default context.) When **\$d-script** is instantiated with details of a particular

solution, i.e. a particular coating, a new child context is created to deal only with that specific solution. So when instantiating **\$d-script** with details of a bitumen coating, the instantiation is done in a new context, bitumen. (Names of contexts will be show bold and underscored.) Thus from within the bitumen context, **\$d-script** can inherit values from **\$d-script** in the root context, while creating and manipulating values in **\$d-script** which are relevant only when considering bitumen coatings. Values established in a child context do not corrupt ancestor context values, thus enabling reasoning in alternative worlds. This is central to the way ADEPTUS handles the task of considering different candidate solutions.

5.2 The Knowledge Base

5.2.1 Overview

Several types of knowledge are relevant to the task of selecting the optimal anti-corrosive coating for a given situation. It must be possible to represent the physical processes at work as a coating is applied, dries, and is put into service. Some representation of the substrate material must be present. Information must be available about individual coatings, whether considered in bulk, as a wet film on the substrate, or as a dry film. Such knowledge may encompass chemical information, visual descriptions,

physical attributes, and the manner in which these and other external factors interact under certain circumstances. This implies that some knowledge about external influences must also be present in the system. Such knowledge includes, for example, descriptions of various chemicals, corrosive environments and mechanical forces which may act upon a coating film. Distinct from the knowledge that contributes to the causal model within the system is a body of information about the system's problem-solving task. This is knowledge of the procedures required to select one of a finite set of discrete solutions. This 'problem-solving script' is maintained independently of the causal model and its associated knowledge base.

5.2.2 Scripts

5.2.2.1 Structure of the scripts

Scripts in ADEPTUS are essentially uninstantiated causal chains. Each script is implemented as a schema in SRL. The domain-level script is called **\$d-script**; the selection-level, **\$s-script**. **\$d-script** has two kinds of slots: state slots and process slots. **\$s-script** has state slots and act slots. (In the remainder of section 5.2.2, it is unnecessary to distinguish between 'acts' and 'processes', so only the term 'process' will be used.)

Each slot has associated with it a 'meta-schema'. A meta-schema holds information about the slot to which it belongs. As the name implies, the meta-schema is itself a schema, and its information is held in slots and slot values. This information can originate from two sources; system defaults and the programmer. Any slot known to SRL has a standard meta-schema linked to it, containing such slots as *range*, *default*, *cardinality* etc. Other slots may of course be added by the programmer. The slots of this meta-schema are described as the 'facets' of the original slot, and will be denoted by the use of a different typeface.

States and processes in the script are fashioned into a causal chain via certain facets. For forward movement in the script, the stative image slots have an *enables* facet, the value of which indicates which processes are enabled by that state. Active image slots have a *forward-link* facet called *results-in*, pointing to the resultant states. Inverse pointers exist. The *enables* facet has an inverse of *prerequisites*, and the *results-in* facet has an inverse of *produced-by*.

Most slots in ADEPTUS have eponymous schemata that are part of the **relation** hierarchy. It is the root of this hierarchy that provides the template for a slot's meta-schema. Thus the **relation** schema itself is a general description of information that should be available to describe any slot

in the system.

Any schema that inherits from the **relation** schema can have a named inverse relation. This inverse relation can automatically be put into place (if an SRL switch is set appropriately) when a slot is created links two schemata. The logical inverse of the *enables* slot is *prerequisites*. Thus, when a link of the form

stateX enables processX

is created, one can envisage this facility being employed so that an inverse link of the form

processX prerequisites stateX

is automatically put into place. Similarly, the logical inverse of *results-in* is *produced-by*, so the creation of the link

processX results-in stateY

would also imply the creation of the reverse link

stateY produced-by processX.

However in ADEPTUS this facility cannot be employed in this way. This is because *enables* is not a slot of the schema **stateX**. Although there is a schema called **stateX**, it is only an eponymous schema for the slot *stateX*, whose presence serves to establish *stateX* in the **relation** hierarchy. So, while *enables* is a slot of **stateX**, it is being used as a facet of the *stateX* slot in the **\$d-script** schema. *enables* is a slot of a meta-schema. The actual relationship is thus

`$d-script stateX <value>`

```
|  
| meta-schema  
|
```

`m00031 enables processX`

`m00031` is a schema created by SRL, and is an instance of the `stateX` schema. Any automatic inverse creation would be between the meta-schema of the state (`m00031`), and the `processX` schema. The inverse links are therefore created by explicit command. This enables movement in both directions along the causal chain.

The domain-level script currently begins with a description of the world before a coating has been applied to the substrate, but it assumes that surface preparation has already been carried out. The elimination of the surface preparation description simplifies the model in two ways. First, the sheer size of the script is reduced; and secondly it means that all active images at the domain level are processes, thereby excluding any need to deal with physical acts. This is because operatives are required to prepare the surface and to apply the paint (whether those operatives are human or robot), but once the paint film is in contact with the substrate, no further intervention by purposeful actors is required. Thus all subsequent active images are processes.

The selection-level (s-level) script describes the actions

of ADEPTUS in carrying out selection from a set of known potential solutions. All its active images are ACTS, and ADEPTUS is implicitly the actor. The actions are manipulations of the d-script; they can be regarded as 'thoughts' of the system, and hence are analogous to 'mental acts' in Schank's work.

5.2.2.2 Active Images

An active image has two quite separate effects when it 'executes', whether at the s-level or the d-level. In order for the system to attempt to carry out, say, the drying process within the model, the enabling state must at the very least be adequately instantiated. If it is, then the execution of the drying process will do two things: firstly, it will instantiate the resulting state. It accomplishes this by creating a binding list to indicate that the resulting state slot has a fixed value. This is the macro-effect of the process. Secondly, it will alter some values in the schemata that constitute the binding for the resulting state. This is the micro-effect of the process. For example, the STICK process is described in the **STICK** schema shown in Figure 16.

All processes in the d-script and acts in the s-script have such an associated eponymous schema.

```

{{STICK
  is-a physical-process
  script-level $d-script
  macro-effects (employ-rules '( <rulebase> )) }}

```

Figure 16

For example, Figure 17 shows a process slot in the \$d-script schema.

```

{{ $d-script
  . . .
  process2 (STICK ?C ?S)
    [instance processslot]
    [results-in state3]
    [prerequisites state2]
    [can-affect ((coating adhesion)) ]
    [affected-by ((substrate constituents)
                  (substrate surface-key)) ]
    [binding-list ((?E . application-environment4)
                  (?C . wet-film-bitumen2)
                  (?S . substrate3)) ] }}

```

Figure 17

The macro-effect operates on the state indicated by the value of the causal link results-in. The way in which this state is instantiated is governed by knowledge held in the value of a slot attached to the STICK schema. The value of this slot must be executable; in all cases thus far in ADEPTUS, it is a rulebase with an interpreter function.

The micro-effect of the process is permitted to alter values within the schemata which constitute the bindings for the resulting state. The choice of slots affected by the process is controlled by a facet of the process slot (see Figure 17). The facet is a can-affect link between the active image in the script and the attributes altered by the process involved in that image. The value of the facet gives the names of the affected schema slot combination in the resulting state. The way in which each schema-slot value is affected is controlled by a 'demon' attached to that slot. Whenever a value is to be created or modified for the slot, the demon uses its rulebase or other executable code, plus the details of the state enabling the current process, to generate the appropriate predicted value for that point in the script.

The positioning of the macro-effect knowledge and the micro-effect knowledge is important. The macro-effect is relevant any time that the process is carried out. If we consider a system where the domain is crop-spraying, a TOUCH process in the domain-level script would have the same macro-effects on its resultant state as it does in the anti-corrosive coating domain. In contrast to this, the actual schema-slot combinations affected would be quite different in the two domains, as would the manner in which they were affected. Thus, the macro-effect knowledge is stored with the process information; whereas the micro-effect knowledge is specific to the schemata that

contain domain-specific knowledge, and is stored appropriately.

Details of individual processes and the manner in which they execute can be found in section 5.3.4.

The can-affect facet of a process slot is a forward link to the resultant state. A reverse link also exists, which shows the elements of the enabling state which are relevant to the micro-effects of the process. This reverse link is the affected-by facet.

```
{{ $s-script
. . .
  process4 (COMPARE ?$D ?IDEAL)
    [instance processslot]
    [results-in state5]
    [can-affect (Divergence) ]
    [affected-by ($d-script ideal)]
. . .
}}
```

Figure 18

5.2.2.3 Stative Images

The value of each state slot in the script schema is a simple representation of the pertinent parts of the total world state; e.g. (?C ?S ?E), meaning that the interesting parts of the total world state are the coating, the substrate and the environment, represented by the tokens

?C, ?S and ?E respectively. At the domain level, this total world state encompasses such diverse schemata as **environment, coating and substrate**. At the selection level, relevant schemata are **\$d-script, Divergence and Ideal**.

In some states, a major part of the world description can be ignored. For example, at the domain level, the state between the TOUCH and STICK processes has no need to refer to the environment. The environment is of course still part of the total world state, so it must be possible to enquire about it at the current point in the d-script even though it does not figure in the state value. This type of ability in ADEPTUS is achieved by the manipulation of a binding-list attached to the state slot. The actual value of the state slot is never altered if the script is running normally; that is, if the process causing the state has not been disabled. The modifications made to the state on instantiation affect only the binding list attached to the slot. Thus a token in the state value is associated with a particular schema only through the binding list, not by substitution of the token. The binding list is inherited from the prior image in the causal chain, and is merely modified as necessary. Thus, although the environment may not be explicitly mentioned in a state, it will still be present in the binding list, and hence available for scrutiny when required.

If, however, the causative process for a state has been

disabled, this may affect the value of the state itself. For example, at the domain level, if the TOUCH process were disabled for any reason, the resultant state value would no longer have the form

```
(PHYSCONT ?C ?S)
```

since there would be no physical contact between the coating (?C) and the substrate (?S). The state value would simply be given as

```
(?C ?S)
```

indicating the existence of both coating and substrate without physical contact.

```
{{ $d-script
```

```
state3 (?ED ?S)
  [instance stateslot]
  [enables process3]
  [produced-by process2]
  [active-enablers ((coating adhesion)
                   (coating flexibility)
                   (coating hardness)
                   (coating single-film-thickness)
                   (environment-description atmosphere))]
  [active-results ((substrate constituents)
                  (substrate surface-key)) ]
  [binding-list ((?E . drying-environment)
                (?C . wet-film-bitumen3)
                (?S . substrate4)) ]
. . . }}

```

Figure 19

```

{{ $s-script
  ...
  state2  (? $D ?SOL)
    [instance  stateslot]
    [enables   process2]
    [produced-by process3]
  ...
}}

```

Figure 20

State slots have an active-enablers facet illustrated in the example in Figure 19. This contains the schema-slot combinations in the state whose values may affect the performance of the enabled processes. The value of this facet maybe regarded as the inverse of the affected-by facet of the enabled processes. Figure 19 illustrates a typical domain-level state representation, and Figure 20 a typical selection level state representation.

5.2.2.4 The Binding List

The binding list is a facet applicable both to process and to state slots. The necessity for its existence was explained in section 5.2.2.3; for any given slot it is a list of pairs of objects, each pair consisting of a token and a schema name. This association of tokens and schemata gives a snapshot of the characteristics of the domain at the current point in the script. It always represents the fullest possible description of the domain, regardless of

the focus of interest of its owning slot. Examples of the binding list in the domain level script can be seen in Figures 17, 18, 19 and 20.

5.2.3 Knowledge about coatings: The set of potential solutions

5.2.3.1 The coating hierarchy

The potential solutions of the selection problem are all nodes of the *is-a* linked hierarchy that has the schema *coating* as its root (See Figure 21).

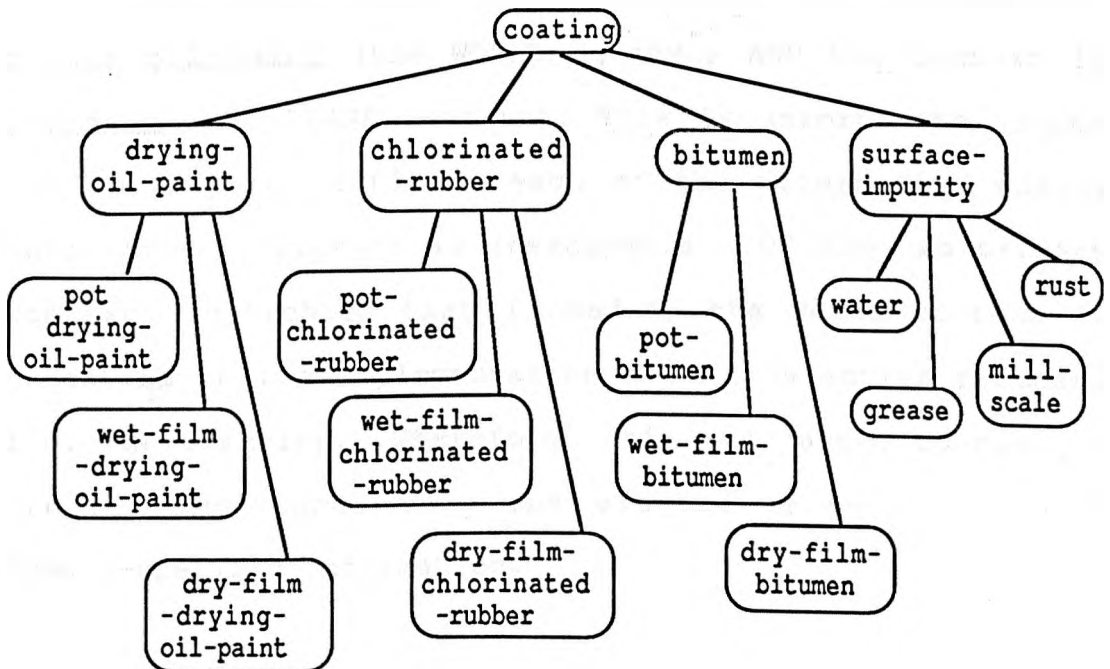


Figure 21

As a coating is applied and hardens, its attributes change.

The representation of the various 'phases' of the coating lifecycle are held as separate subclasses of the coating. This is not an entirely satisfactory solution to the problem. An earlier solution was to employ SRL's context capability, defining pot, wet-film and dry-film as contexts. From these contexts, one could view a particular coating and only perceive the attributes relevant to that context. This seems a more elegant solution than the somewhat cumbersome *is-a* link hierarchy. However, the use of contexts in describing domain-level knowledge precludes the use of contexts for reasoning in alternative worlds. It is not possible in SRL for a child context to inherit through more than one path to the root-context; that is, one cannot ask about a state where the context is drying-oil-paint (the WORLD context) AND the context is dry-film (the PHASE context). This is unfortunate, since the ability to partition each of the alternative worlds into various contexts is inescapable. Of the two primary context hierarchies that I needed, the world context is essential to the implementation of the selection process, i.e. the s-script. Therefore, this was judged to carry a greater importance than the elegant representation of domain-specific information.

The slots of the leaf nodes in the coating hierarchy are available to the leaf node schemata by inheritance. However, they are not inherited from within the coating hierarchy, but from another hierarchy, of which the coating

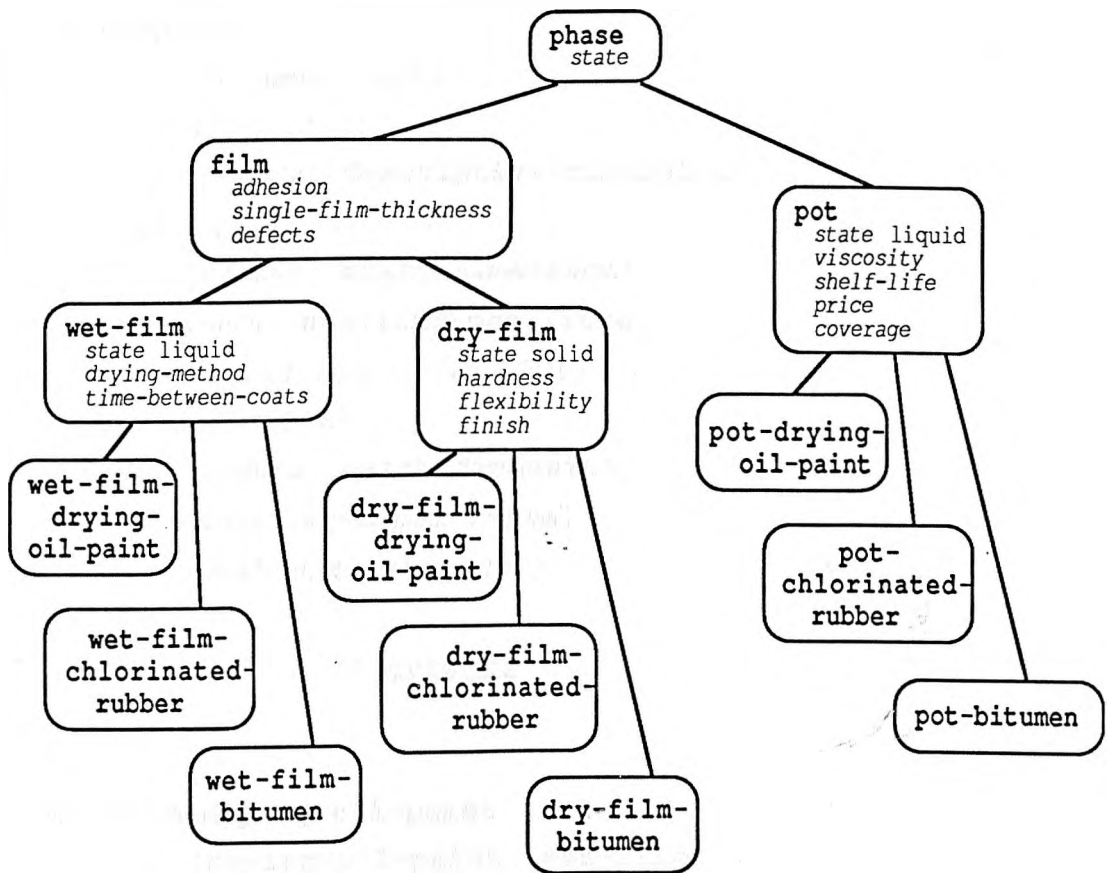


Figure 22

leaf nodes are also nodes. This hierarchy is the **phase** structure described below. Thus, the ability of a schema to belong to multiple schema hierarchies has been substituted for the requirement for inheritance from multiple contexts.

5.2.3.2 The phase hierarchy

Regardless of the particular coating used, a film of

```

{{pot-bitumen
  is-a (bitumen pot)
  viscosity (2 3)
    [instance descriptive-dimension]
  price (4 8)
    [instance exact-dimension]
    [units sterling-per-litre]
    [valid-range (0 100)]
  coverage (5 8)
    [instance exact-dimension]
    [units sq-m-per-litre]
    [valid-range (0 1000)]
}}

```

Figure 23

```

{{wet-film-drying-oil-paint
  is-a (drying-oil-paint wet-film)
  drying-method oxidation
    [instance feature]
  time-between-coats (0.5 0.7)
    [instance exact-dimension]
    [units log-hours]
  state liquid
    [instance feature]
  adhesion
    [instance descriptive-dimension]
  thickness
    [instance exact-dimension]
    [units microns]
  defects
    [instance feature]
}}

```

Figure 24

```

{{dry-film-drying-oil-paint
  is-a (drying-oil-paint dry-film)
  state solid
      [instance feature]
  hardness
      [instance descriptive-dimension]
  flexibility
      [instance descriptive-dimension]
  finish gloss
      [instance feature]
  adhesion
      [instance descriptive-dimension]
  thickness
      [instance exact-dimension]
      [units microns]
  defects
      [instance feature]
}}

```

Figure 25

material has characteristic attributes. The same can be said of a coating in bulk in the pot. It is these characteristic attributes which are bequeathed in the **phase** hierarchy. (See Figure 22.) Using the slots inherited through this structure, the different phases of a coating can be fully described. Figures 23, 24 and 25 illustrate the description of the three phases of a drying oil paint.

At the intersection of the **coating** and **phase** hierarchies there exist schemata containing values specific to a coating, with slots appropriate to the phase under consideration.

Each slot has several facets associated with it. Some of these are inherited from the eponymous schema connecting the slot into the **relation** hierarchy. These facets are discussed further in section 5.2.5.2 below. An SRL-defined facet of great importance is also associated with each slot, although it has not been included in previous diagrams for the sake of clarity; this is the **demon** facet. It contains the name of zero or more demon schemata, which define any action to be taken when the value of the slot is under discussion. The demon schema controls the type of slot access for which the demon will be roused (e.g. value retrieval or value creation) and the point at which the demon will act relative to the time of the access. Thus a demon may be aroused before or after accessing the value.

The demon facility in SRL is intended for use in reactive processing; that is, when a slot value takes on a certain value another slot may need to change its value in a specific manner. (The example supplied by the creators of SRL is that when the **fido mood** slot takes the value 'happy', then the **fido tail** slot should immediately be set to 'wagging'.) However, in ADEPTUS the demon schemata are employed as repositories for specialist knowledge bases that are competent to predict the associated slot's value, depending on the world state that obtains when they are invoked. In this sense they are comparable to Minsky's 'agents' [Minsky 1979].

5.2.3.3 The Ideal Film

There is within ADEPTUS an **Ideal** schema, which characterises the perfect solution to all corrosion problems. This description is an agglomeration of the best attributes of all the coating films (i.e. solutions) known to the system. Thus the ideal film is flexible, hard, and extremely well adhered to its substrate. The value of +3 given to any **descriptive-dimension** slot is the maximum possible. The film is impervious to extremes of temperature and to any chemical spillage, and capable of sustaining severe mechanical damage without loss of integrity: this is reflected in the value **solid** for the *state* slot, and the nil value in the *defects* slot. It is made from a cheap bulk phase; the value in the *price* slot is that of the cheapest coating known to the system.

Certain slots within the **Ideal** schema do not have categoric values assigned to them. In Figure 26 these are the *thickness* and the *finish* slots. The ideal thickness of an anti-corrosive coating depends on two factors; the coating under discussion and the desired effective lifetime of the coating. The diagram in Figure 27 illustrates this. The nature of the coating can be isolated from the context in which the **Ideal** schema is employed. The desired lifetime is a preference on the part of the user. (I say preference rather than requirement, since it is just such factors

```

{{Ideal
  state solid
  hardness +3
  flexibility +3
  finish
  adhesion +3
  thickness
  defects nil
  price 1.5      }}

```

Figure 26

which may be compromised in the face of other considerations, notably cost.) The ideal *finish* of a coating is also a preference on the user's part. If the user wants a high-gloss finish, then that is the ideal. If the user indicates indifference to the finish obtained, then no ideal value is available. The *thickness* and *finish* slots do not contain values, but have attached demons that provide a mechanism for obtaining the ideal value in the current situation.

The **Ideal** schema does not actually belong to the physical domain, since it has no correspondence with a real physical entity. It can better be thought of as a figment of ADEPTUS' imagination; a mythical entity against whose advantages any candidate recommendations will be judged. It is a necessary construct for the problem-solving capabilities of ADEPTUS when a selection task is being undertaken. It is described here because of its participation in the **phase** hierarchy. Although the **Ideal**

schema is supplied to ADEPTUS as part of its initial knowledge base, it could in fact be omitted. ADEPTUS could include sufficient knowledge for the construction of the

Minimum Film Thickness

(microns)

time to first maintenance coating	<5 years	>20 years
drying oil paint	85	230
chlorinated rubber	120	300
bitumen	250	500

Figure 27

Ideal schema from the best possible dry film phases of the potential solutions in the knowledge base. The knowledge about non-agglomerated ideal values (i.e. *thickness* and *finish*) would of course need to be supplied to ADEPTUS to be incorporated in the schema.

5.2.4 Knowledge about the Substrate

5.2.4.1 Substrate representation

A hierarchy exists in ADEPTUS having the schema **substrate**

as its root. This hierarchy is illustrated in Figure 28.

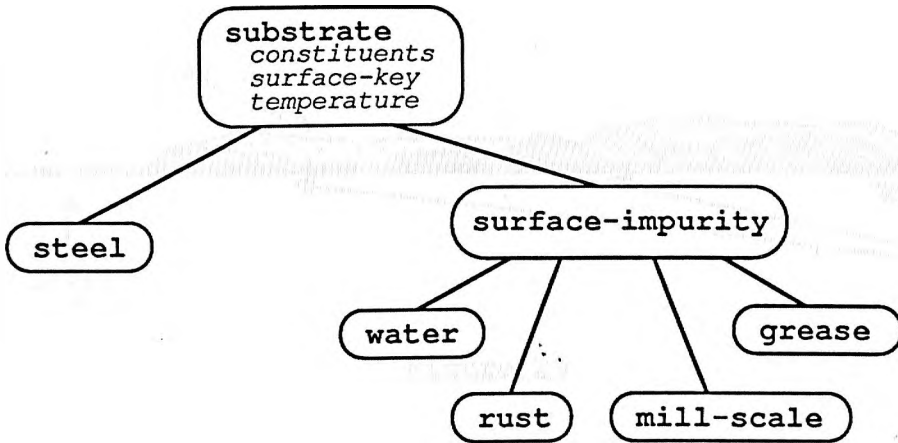


Figure 28

Clearly, it must be permissible to nominate a surface impurity as a substrate; very few substrates will be entirely free from impurities of some sort. However, Figure 21 shows that the **surface-impurity** schema is also a node in the **coating** hierarchy. This enables the system to view a surface impurity as a substance deposited on the substrate at some past time and thus susceptible to removal.

When a problem description is formulated, a substrate description must be created. A major representational problem was encountered here; that of how to describe a physical three-dimensional arrangement in simple symbolic fashion. An example of the physical reality of the

substrate is depicted in Figures 29 and 30.

Cross Section

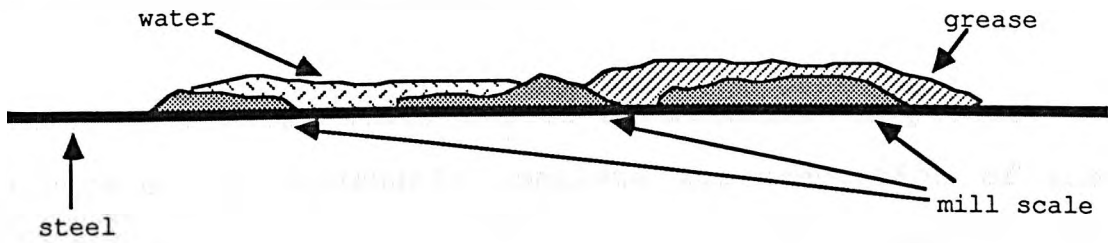


Figure 29

Plan View

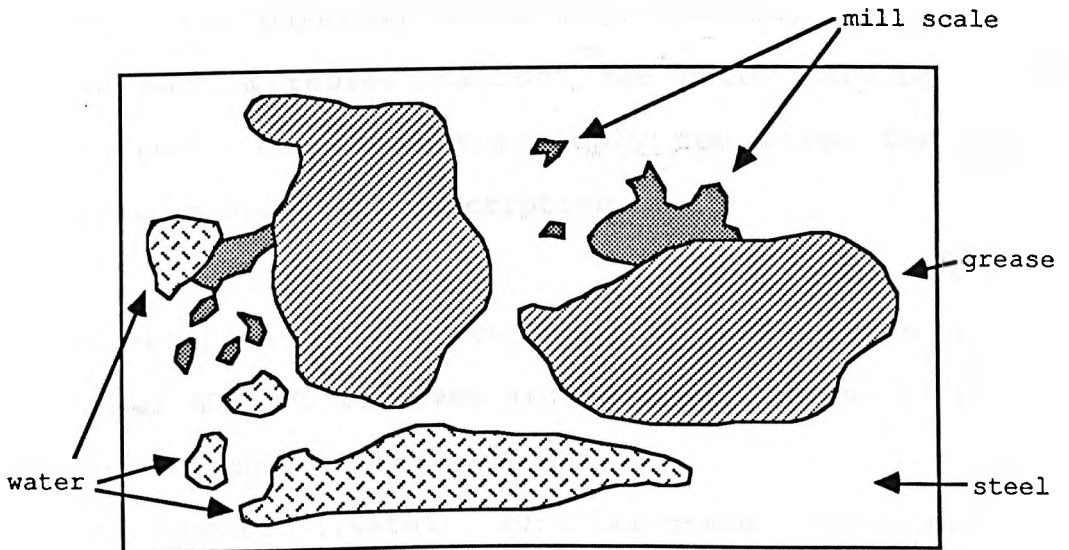


Figure 30

Obviously such an arrangement has many attributes. From Figure 29 one can nominate for consideration the substances involved, their relative positions in the 'sandwich'

between essential substrate and environment, and the thickness, adhesion etc of each layer. From Figure 30 it is apparent that the composition, area and distribution of each uppermost layer should be noted.

These attributes would indeed be required to produce an accurate and reasonably complete representation of the substrate-system, but they are not all necessary to ADEPTUS' purpose. First, one could not reasonably expect such detailed information to be provided by the putative user of the system. So from that viewpoint a substrate representation of such complexity is wasted. Secondly, the domain knowledge contained in ADEPTUS does not predict events in the physical world with accuracy sufficient to utilise such a representation. The error margins of the domain-level inferences are simply too large for such a fine-grained substrate description.

The description of the substrate in use in the current version of ADEPTUS revolves around the *constituents* slot in the created schema. For example:

```
constituents ((water . 40) (mill-scale . 30) (steel . 50))
```

The value of the slot is an association list. The last element in the list represents the essential substrate. Any overlying layers are represented by elements added to the head of the list. The first element then shows the outermost layer of the composite surface.

Each element in the a-list is composed of a schema name as the key and a number in the range 0 to 100 as its associated value. The numerical value indicates the percentage of essential substrate covered IN TOTAL by its key substance. The obvious exception to this is the pair whose key is the essential substrate itself. In this case, the numerical value indicates the percentage area exposed. It is clear from the example that substances overlying the essential substrate may overlap partially, completely or not at all. In the example, the water (40%) and mill scale (30%) must overlap to some extent to give a value of 50% bare steel.

The two remaining slots in any substrate schema are *surface-key* and *temperature*. The former is a dimension whose value is directly related to the standard of surface preparation decided upon by the system user. The latter is also, of course, a dimension. It gives the temperature of the substrate itself, as distinct from the environmental temperature. This may be important when the ambient temperature is within a climatically normal range, but the substrate is maintained at an abnormally high or low temperature.

5.2.5 Associated Entities

5.2.5.1 Liquids

There exists within ADEPTUS a hierarchy with the schema **liquid-substance** as its root. This hierarchy is illustrated in Figure 31. Four major subtrees are evident, having **chemical**, **oil**, **resin**, and **water** as their root schemata.

The **chemical** subtree is present for use in describing spillages on the coated surface. The leaf nodes of the initial knowledge base hierarchy can inherit the *pH* slot from the **chemical** schema, but no value is placed in this slot. A restriction on the values which may be used for the *pH* slot is specified at each leaf node by using the SRL-defined range facet. This enables the specification of (in this case) a piece of Lisp code which will be used to test any submitted value. Thus it is possible to restrict the **acid** *pH* values to between 0 and 7, and the **alkali** *pH* values to the range 7 to 14. (A perhaps surprising omission in the description of the chemicals is the absence of a *chemical-formula* slot. This was a part of the original design of this tree, but it quickly became obvious that to employ symbolic representations of chemical reactions would be beyond the scope of the current research.) Values for the slots in the specific chemical schemata are supplied by the user. Thus, if a likely spillage is weak hydrochloric acid, a child schema of

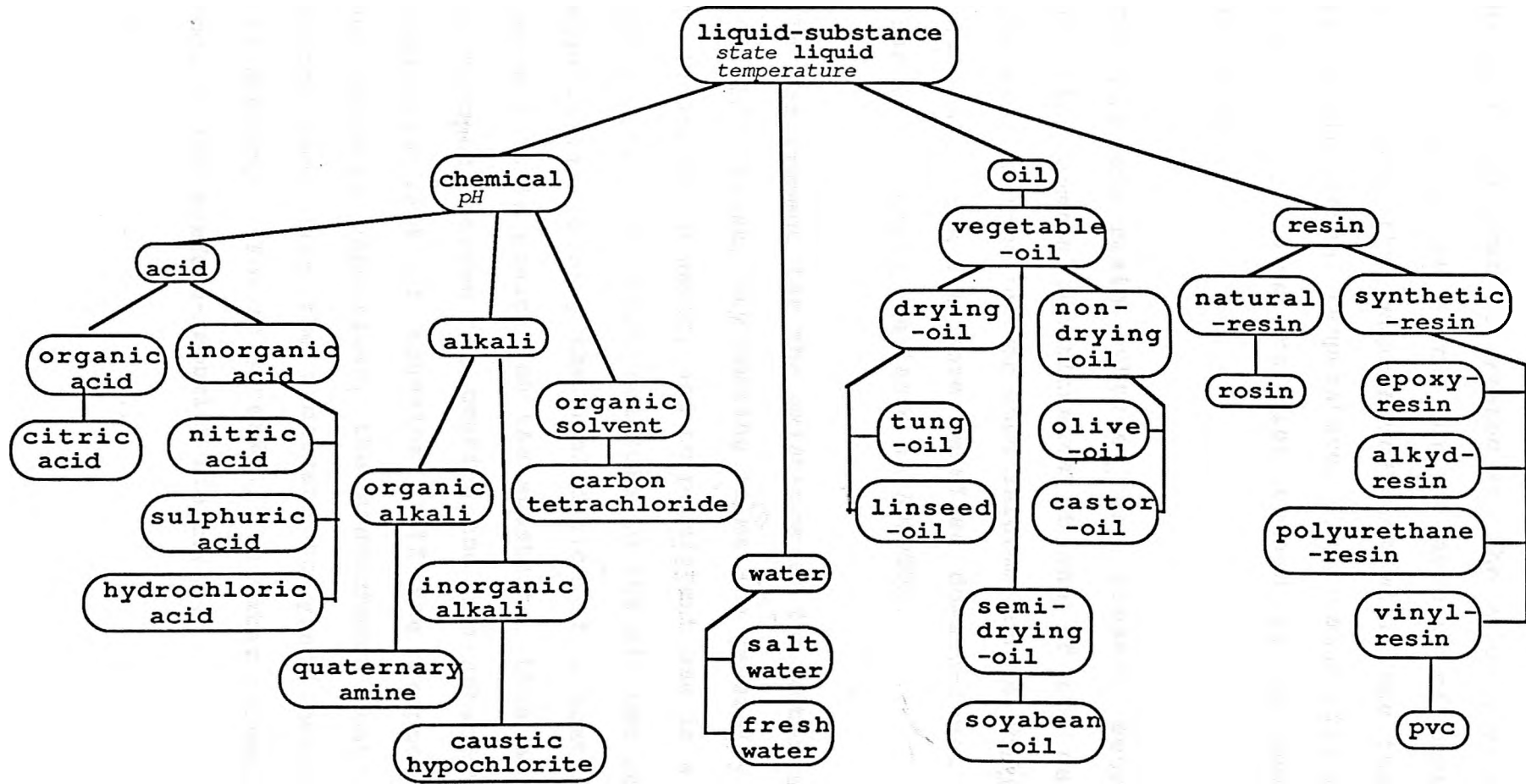


Figure 31

hydrochloric-acid is created and the value 5 or 6 put into the inherited *pH* slot. In the absence of contradictory information, the *temperature* slot will take the value of the environmental temperature. This value will not be put into the *temperature* slot unless it is specifically accessed.

The **oil** and **resin** subtrees are present because they describe common constituents of one of the major paint classes. Active use of such information in the predictive sense will require more detailed domain-level knowledge than is currently available in ADEPTUS.

The requirement for the existence of the **water** subtree is twofold. First, dry coating films are generally sensitive to immersion in water, and to persistent use in a splash or spray zone. Secondly, moisture in the air can be of great significance during the application of a coating. If a water film is present on the substrate, this moisture can be trapped between the coating and the substrate causing localised lack of adhesion. If the trapped moisture subsequently vapourises, the non-adhered coating can be forced away from the substrate to yield characteristic 'blistering'. For this reason, the **water** schema is also a node in the **surface-impurity** hierarchy.

5.2.5.2 The relation hierarchy

As previously stated, all slots in ADEPTUS are nodes in an *is-a* hierarchy with the SRL system schema **relation** at its root. This hierarchy can conveniently be discussed in two separate parts.

The first part is the tree of relations concerned with script definition, and is illustrated in figure 32a. These relations are employed in scripts at either level, since the scripts are intended to function in essentially the same way regardless of the script domain. The script definition relations fall naturally into two categories, script slots and script facets. The meaning and function of these relations was described in section 5.2.2.

The second part of the **relation** tree is shown diagrammatically in figure 32b.

The leaf nodes of the **feature** and **dimension** subtrees are relations specific to the physical domain of expertise. Domain-level attributes can be categorised on the continuous/discrete quality of the set of possible values for those attributes. A **dimension** has a continuous set of values, whereas a **feature** has a discrete set. This was discussed in section 3.5.2. The control of the range of possible values for features is undertaken using SRL's system-defined **range** facet for the slots. Thus it is

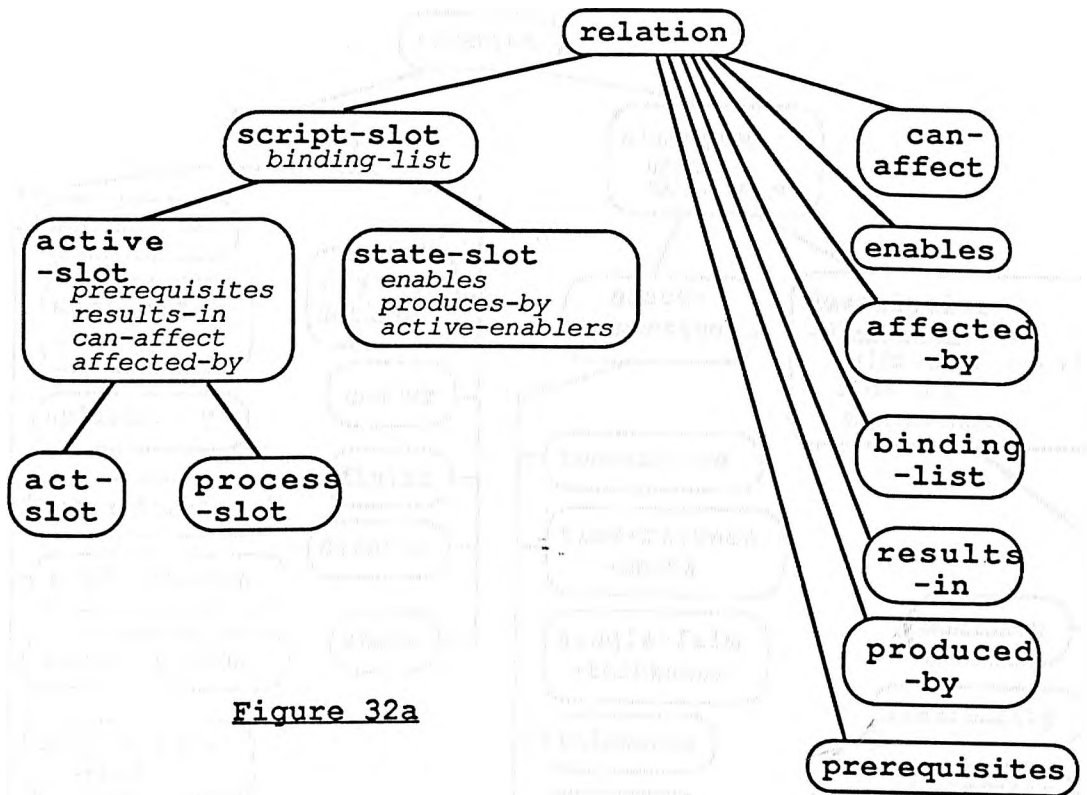


Figure 32a

possible to restrict the value of an *immersion-in* slot to some entity within the **liquid-substance** hierarchy, or the value of *mechanical-damage* to be an instance of either the **abrasion** or the **impact** schema.

The dimensions known to ADEPTUS are divided into exact dimensions and descriptive dimensions. Different facets are attached to each class.

The exact dimensions are those given a numeric value which has some physical significance. Examples in ADEPTUS are *temperature* and *price*. These dimensions are illustrated in Figure 33.

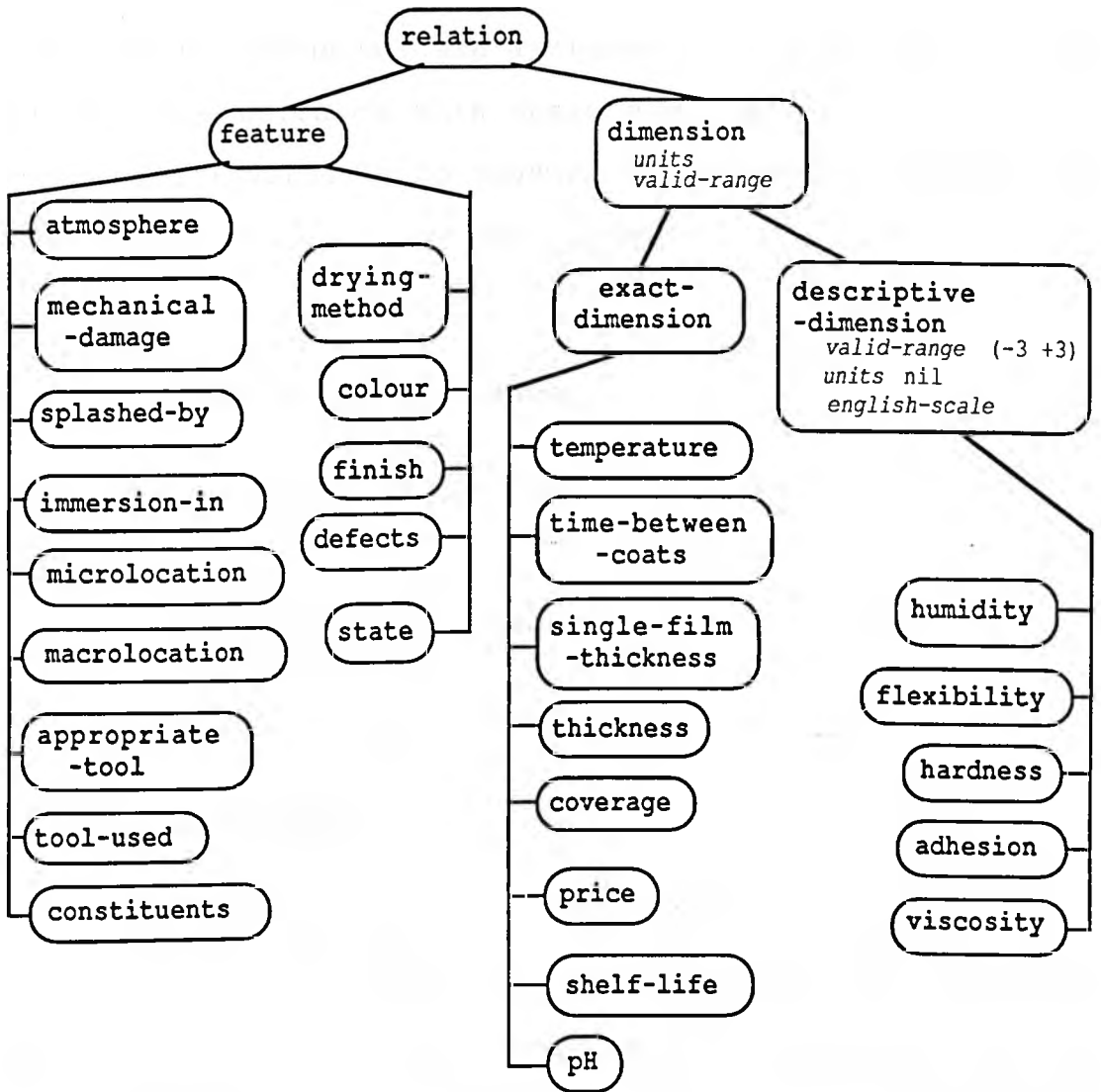


Figure 32b

```

{{temperature
  is-a exact-dimension
  units degrees-C
  valid-range (-100 600)  }}

{{price
  is-a exact-dimension
  units sterling-per-litre
  valid-range (0 100)  }}
  
```

Figure 33

Descriptive dimensions are assigned rank values on a fixed scale. The units of this scale are imprecise and do not correspond rigorously to measurable values in the physical world. For example, consider Figure 34.

```
{{adhesion
  is-a descriptive-dimension
  valid-range (-3 +3)
  english-scale (strong weak)  }}
```

```
{{viscosity
  is-a descriptive-dimension
  valid-range (-3 +3)
  english-scale (thick thin)  }}
```

Figure 34

This shows the dimensions of adhesion and viscosity; both are experimentally measurable quantities with standard units. However, the use of such units is once again at too fine a grain size for ADEPTUS. Neither the information provided by the user nor the accuracy of the inferences require such details. The descriptive dimensions have associated with them a slot *english-scale* which enables the value of the dimension to be translated into English in a simple way. Qualifiers are associated with the modulus of the integer part of the value of the dimension as follows.

INTEGER	QUALIFIER
1	fairly
2	very
3	extremely

These qualifiers are used as a prefix to the appropriate adjective taken from the *english-scale* slot; the first element of the list which is the slot value corresponds to positive values of the dimension, the second to negative values. A zero value has no prefix and always corresponds to the word 'average'. For example:

adhesion -1 => *adhesion* fairly weak

viscosity 2 => *viscosity* very thick

All the dimensions known to ADEPTUS could be represented uniformly; all as exact dimensions, or all descriptive. The criterion for usage of one category in preference to the other is a purely pragmatic one. The considerations that led to the assignment shown in figure 32b are the terms used by the human expert and the level of detail that he or she employs.

The remaining subtree of the **relation** hierarchy, **feature**, is also shown in the Figure 32b.

5.3 Performing the Selection task

5.3.1 Introduction

The following subsections describe the steps taken by ADEPTUS in trying to make a selection of an anti-corrosive coating. The structure and meaning of the information supplied by a user of ADEPTUS is described. The workings

of the s-script and the d-script are then explained, focussing on the active images and their effects on the 'mental state' of ADEPTUS.

5.3.2 Situation Specific Knowledge

5.3.2.1 Overview

The surface characteristics of the user interface are not of interest in this research. I have therefore omitted from the s-script any input/output acts which would be necessary for a satisfactorily complete system. The user's input is viewed as instantiated schemata already in a form suitable for use by ADEPTUS.

The information from the user must encompass a description of the substrate when ready for coating, and characteristics of three distinguishable environments:

- the conditions obtaining when the coating process takes place
- the conditions obtaining during the drying of the coating
- the likely environment for the coating during its effective lifetime.

The third of these includes information on possible chemical spillages and mechanical damage to the coating. Such information must be available to ADEPTUS before the

problem-solving process can begin.

5.3.2.2 The substrate

Knowledge about the substrate is held in a child of the **substrate** schema described in section 5.2.4. Both the *constituents* slot and the *surface-key* slot values will be influenced by the preparation carried out on the surface. If, as is presently true, no representation of preparation exists in the causal model employed by the system, a simple translation can be made between a standard surface preparation (e.g. Sa3, Sa2 in the Swedish Standard) supplied by the user and the appropriate values in the internal schematic representation of the substrate.

5.3.2.3 The Environment

Three separate environments are of interest in the problem of selecting an anti-corrosive coating. The first two of these refer to the environment in which application of the coating is carried out and the environment in which drying occurs. Three essential factors are included in the **application-environment** and **drying-environment** schemata: the atmospheric humidity, the ambient temperature and the 'microlocation'. The microlocation shows whether the substrate is indoors or outdoors, sheltered from rain or exposed to the elements. The *microlocation* slot is a

feature; both of the other slots are dimensions. See Figures 35 and 36. In these examples, instances of **application-environment** and **drying-environment** have been created to hold the user's situation description. **application-environment** and **drying-environment** are both subclasses inheriting from the **environment-description** schema.

```
{{application-environment1
  instance application-environment
  humidity (-3 1)
    [instance descriptive-dimension]
  temperature (10 20)
    [instance exact-dimension]
  microlocation indoors
    [instance feature]
    [range (or indoors outdoors) ]    ]}}
```

Figure 35

```
{{drying-environment1
  instance drying-environment
  humidity (-3 1)
    [instance descriptive-dimension]
  temperature (10 20)
    [instance exact-dimension]
  microlocation indoors
    [instance feature]
    [range (or indoors outdoors) ]    ]}}
```

Figure 36

The values for *temperature* and *humidity* are ranges of user-estimated values. The information coming from the user is in the form of predictions of future environments. Certainly in the anti-corrosive coating domain, where factors such as weather can influence choice, it is not feasible to expect the user always to be categoric about the expected environment.

Often, as in the example, the application and drying environments will be identical, but this is not necessarily the case.

The most interesting of the environment descriptions is the **corroding-environment** schema. This is shown in figure 37. The three slots from the application/drying environments are still relevant, but several other factors now come into play.

The syntax of SRL's *range* facet allows a wide range of types of restriction. The or used in the *microlocation* and *macrolocation* slots requires that the slot value be one of the schemata names given. list demands an ordered list conforming to the restrictions described in its arguments. Thus, acceptable values for *macrolocation* would be (**rural inland**), or (**industrial coastal**) etc. The type restriction is characteristic of SRL. It requires that a value should bear a certain relation to another schema. For example, the value of the *immersion-in* slot must be a

```

{{corroding-environment
  is-a environment-description
  humidity
    [instance descriptive-dimension]
  temperature
    [instance exact-dimension]
  microlocation
    [instance feature]
    [range (or indoors outdoors)]
  macrolocation
    [instance feature]
    [range (list (or rural industrial)
                (or coastal inland)) ]
  splashed-by
    [instance feature]
    [range (set (type is-a liquid-substance)) ]
  immersion-in
    [instance feature]
    [range (set (type is-a liquid-substance)) ]
  mechanical-damage
    [instance feature]
    [range (set (type is-a mechanical-damage-description))]
}}

```

Figure 37

is-a liquid-substance. This relationship need not be direct; it is quite acceptable for the link to be via an inheritance pathway. So, although **salt-water** is a grandchild, not a child, of **liquid-substance**, the *is-a* link permits inheritance, and the range restriction on the *immersion-in* slot would be satisfied with the value **salt-water**. set allows values which are unordered lists of zero or more elements, all of which conform to the

restrictions given as arguments. For example, *splashed-by* could have a value of

(**citric-acid organic-solvent linseed-oil nitric-acid**)

or (**fresh-water**) or ().

The *macrolocation* slot describes in general terms the location of the substrate. This may be rural or industrial, and coastal or inland. The value here is concerned with the degree of pollution or corrosiveness in the atmosphere.

The *immersion-in* slot must have as its value an element of the **liquid-substance** hierarchy. The *splashed-by* slot may have multiple values, since many substances may be spilled on the substrate during the lifetime of the coating. For both of these slots' values, if the liquid substance concerned is a member of the **chemical** subtree, the actual schema will be a specially created child-schema of one of the existing leaf nodes of the **chemical** subtree. The values in this child-schema's slots can then be specified by the user. The obvious example of this is in defining the pH of a spilled chemical; a coating may be eminently suitable for protection against weak acids but not against strong acids.

The remaining slot in the **corroding-environment** schema is *mechanical-damage*. The acceptable values for this slot are user-instantiated child-schemata of either **abrasion** or

impact, which are shown in Figures 38 and 39.

```
{{abrasion
  is-a damage
  severity
  frequency
  participants      }}
```

Figure 38

```
{{impact
  is-a damage
  momentum
  frequency
  area              }}
```

Figure 39

5.3.3 Running \$s-script

5.3.3.1 Initial Instantiation

When the situation-specific information from the user has been set up as instances of the appropriate environment and substrate schemata, these created schema names are collected up into an instance of the **situation-description** schema. An example is shown below in Figure 40.

This central storage of the problem description information ensures that the tokens used in the state and act slots of the s-script (i.e. ?S, ?EA etc.) are independent of the

```

{{situation-description1
  instance situation-description
  ?S  substratel
  ?EA application-environment1
  ?ED drying-environment1
  ?EC corroding-environment1      }}

```

Figure 40

domain of the d-script. In instantiating the s-script, the problem description is taken from the schema, not from information tied in to the d-script.

The other two schemata needed by the s-script are a copy of **\$d-script** and a copy of the **known-solutions** schema. The latter is illustrated in Figure 41.

```

{{known-solutions
  schema-names (bitumen drying-oil-paint
                chlorinated-rubber)
                }}

```

Figure 41

The SRL schema-copying facility is not implemented in the version available to me, so instead of copies of the required schemata, child schemata are used. These simply inherit all slots and values unchanged from their respective parents. The knowledge base schemata are not directly used; this maintains their integrity and leaves open the possibility of re-running the s-script without losing information from previous executions.

The binding-list facet of the *state1* slot in *\$s-script* now has its value instantiated. If the three schemata described in the previous paragraph are *situation-description1*, *\$d-script1* and *known-solutions1*, then the binding-list facet will take the value

```
((?SD . $d-script1)(?SOL . known-solutions1)
  (?SIT . situation-description1))
```

5.3.3.2 The MERGE act

The MERGE act uses the information provided by the *situation-description* schema to alter the binding list of the initial state of *\$d-script*. It establishes in the domain-level causal model the problem characteristics for this particular problem-solving exercise. This is a micro-effect of the act, and as such is effected through the can-affect facet of MERGE, and the demon attached to the *state1* slot of *\$d-script*.

5.3.3.3 The SHATTER act

Having attached the problem description to *\$d-script*, the next step is to attach a possible solution. This requires that a copy of the partly-bound *\$d-script* schema is available for each possible solution. It is the SHATTER act which accomplishes this proliferation. No new schemata

need to be created by SHATTER; it is at this point in the selection script that the SRL context facility is employed.

The SHATTER act creates a context for each known solution. Within this context it alters the binding list of the *state1* slot of the **\$d-script** schema so that the corresponding solution is incorporated. From this point in the s-script, the **\$d-script** schema in the root context is not used; all necessary manipulations happen to the d-script in one of the solution-specific contexts.

Once all this has been accomplished, the **\$d-script** schema can simply be used in the binding list of the resulting state of the s-script, in the knowledge that all the alternative worlds of interest are accessible via the **\$d-script** schema in contexts defined by the solution schemata names.

5.3.3.4 The RONEACH act

This act consists of the repetition of a sequence of events; performing them once for each solution in its binding list. The name of the solution schema is the same as the corresponding context name; so for the **bitumen** solution, the appropriate context is bitumen. The context for each solution is asserted in turn, and the RUN act is executed on **\$d-script**. In common with the SHATTER act,

only the **\$d-script** schema in the enabling state's binding list needs to be named in the resulting state's binding, the details existing only within the solution contexts.

The RUN act which executes the d-script is the same act which executes the selection script itself. It is an act which pursues the *enables* and *results-in* links in its object script, forcing any detected acts or processes into execution. The RUN act itself does not alter the current context, which is therefore asserted before RUN is invoked.

5.3.3.5 The COMPARE act

A **Divergence** schema is created by the COMPARE act for each solution under consideration. A **Divergence** schema has the same slot names as the final versions of the schemata which represent the essential outcome of the d-script, but the meaning of the value for each slot is quite different. In the anti-corrosive coating domain, this 'outcome' schema will be the ultimate **dry-film-** version of each coating. The slot values of the **Divergence** schema now represent how 'good' or 'bad' that value is judged to be, whereas in the corresponding final **dry-film-** schema the value is a prediction of the physical nature of that attribute, e.g. *thickness 2, defects (pitting blistering)*.

Thus, although the final **dry-film-** schema slots may be a

mixture of features and dimensions, all the **Divergence** slots are dimensions. In fact, they are all values of the same dimension, that of **value-judgment**.

The **Divergence** schema slot values are created by examining each slot of the final **dry-film-** schema and comparing that slot value with an optimal slot value obtained from the **Ideal** schema. The value then attributed to the **Divergence** slot is a measure of the distance between the predicted value for the coating and the **Ideal** value. This is the working definition of the **value-judgment** dimension.

5.3.3.6 The RANK act

A method for ranking the **Divergence** schemata has not yet been settled in detail. The *general-effects* slot of the **RANK** schema (the eponymous schema describing the RANK act) is straightforward. The RANK act will create a child of the **Rank** schema (see Figure 42) conditional only upon the enabling state of the act being adequately instantiated.

```
{{Rank
  ranked-schemata
  suitability          }}
```

Figure 42

The demon attached to the **Rank suitability** slot will then

create a list of values for that slot using a rulebase. It is this rulebase that remains undefined. However, the information that the rulebase will use is established. The rulebase belonging to the **Rank suitability** demon will consider each of the **Divergence** schemata as a conceptual histogram. This is illustrated below in Figure 43.

Example Divergence Schema Histogram

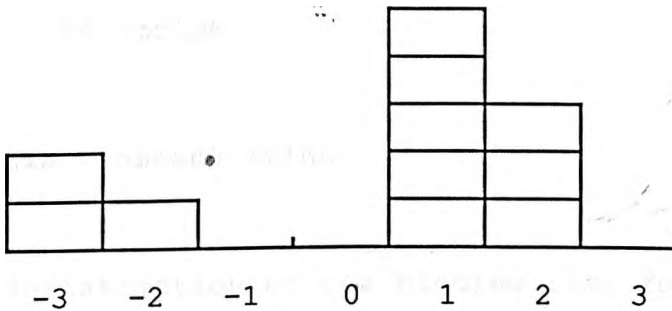


Figure 43

This example shows a Divergence schema where:

- 2 slots have values of -3
- 1 slot has a value of -2
- 5 slots have values of +1
- 3 slots have values of +2

The rulebase needs to be able to decide on the basis of this histogram the overall suitability of the corresponding solution. Clearly, a solution having more than one 'extremely' unsuitable attribute (i.e. slot value of -3) would itself be 'extremely' unsuitable. However, the knowledge required to obtain an overall view from the histogram in other cases is not currently available to the

system; it will be necessary to reconsult the domain experts to obtain this. In this type of expert task, it is likely that a different collection of heuristics will be elicited from each expert. The analysis of the histogram may also depend upon the user's view of the problem, and the costs and risks involved for them.

5.3.4 Running \$d-script

5.3.4.1 Initial Instantiation

The initial instantiation of the binding list for **\$d-script** is carried out by the MERGE and SHATTER acts of the selection script. The solution schema binding is only described within solution contexts, as described in section 5.3.3.3. The appropriate solution context is therefore asserted before the d-script is executed.

5.3.4.2 The TOUCH process

The first process in the current version of the domain-level script is TOUCH. This establishes a physical contact between the coating and the substrate; borrowing the term 'PHYSCONT' from Schank, the resulting state is

PHYSCONT coating substrate

This does not attempt to give any information on the nature of that contact; the coating may or may not attach to the

substrate. The description of the relationship between the two is only elucidated in the next act, STICK.

5.3.4.3 The STICK process

The STICK process can be disabled by the presence of certain elements in the substrate constituents, for example a grease film on the surface of the substrate. Should this be the case, the rulebase defining the macro-effect of the process can alter the structure of the resulting state, so that the coating and the substrate are both present, but not in physical contact.

If the process is not disabled in this way, a binding list is attached to the resulting state, and the process's micro-effects are executed. For this process, it is interesting to note how the prior value of the **substrate constituents** slot has an effect on the resulting value of the same slot. This is a more detailed reflection of the macro-effects of STICKing. If grease is present, the description of the substrate does not change. If the substrate is clean, however, the coating becomes part of the description of the substrate constituents. A suitable term involving the token representing the coating schema (i.e. ?C) is added to the front of the list describing the constituents. It is for this reason that on the macro-level, the resulting state does not need to mention ?C, the token for the coating, if the STICK process has

been successfully carried out. The token is employed in preference to the schema name so that as the description of the coating film changes during the remainder of the d-script, those changes will automatically be reflected within the substrate constituents description.

5.3.4.4 The STATECHANGE process

The nature of the state change is determined by the *drying-process* slot in the coating schema. (The change of state may take place by evaporation, by curing or by a combination of the two.) STATECHANGE has only to instantiate the binding of the coating token in the resulting state and to permit the micro-effects to occur. No interference with the value of the **substrate constituents** value is required, for the reasons explained in the previous section. However, part of the macro-effect of STATECHANGE is that the coating schema in the resulting state is a child of the **dry-film** phase. This must be taken into account when creating the coating successor schema for the resulting state.

5.3.4.5 The WEAR process

The macro-effects of the WEAR process are very simple. A successor coating schema is created and attached to the resulting state binding list. The micro-effects of the WEAR process are of course the most complex in the d-script,

reflecting the variety and complexity of the factors which affect a coating film when it is in use.

5.4 Summary

The current implementation method for the selection script and the domain script have been described. In the next chapter, we return to the central theme of question-answering, and the categorisation of questions in a system based on a causal model.

6.1 Introduction

Wendy Lehnert's work on question answering was carried out as part of the Yale natural language understanding project in the 1970's. The domain used by the Yale workers was simple, non-technical stories about ordinary people. The ideas underlying Conceptual Dependency, causal chaining, scripts etc, have a wide applicability, but if they are to be used in Expert Systems, some profound modifications are required.

6.2 A Reappraisal of Lehnert's Classification

6.2.1 Introduction.

While Lehnert's classification of question types works well enough for the domain of simple news stories, it is not entirely appropriate to the discussion of problem-solving behaviour nor to industrial and technical processes. This section first examines the need to add, modify or abandon individual categories from Lehnert's work. Having obtained a revised list suitable for the domain, a more detailed classificatory structure for question types is described, based on the fundamental script-manipulative acts described

in Chapter 3. It is important to understand where the HOW and WHY questions in conventional expert systems fit into this structure, and section 6.2.4 provides an analysis of this. The last part of this section then highlights the applicability of this analysis. It illustrates how, when people consult experts in reality, they ask questions which span the question hierarchy described.

6.2.2 Modifications to individual categories

Only one of Lehnert's existing categories becomes completely irrelevant in the changed domain: the request category. There does not appear to be any reason to phrase a request to a computer system as a question; it would naturally be phrased as a command.

Another class, disjunction, can be usefully metamorphosed into a class of 'comparison' questions, which have the form

Which has more of attribute X, entity A or entity B ?

This class is more than a disjunction of two confirmatory questions. Since entities A and B have an attribute in common, yet each has an individual value for that attribute, the question class cannot be simple slot-filling. It is proposed, then, that these Comparison questions replace the Disjunctive category.

One extra class of questions should be added to the scheme.

This is the definition class. This encompasses all requests for definitions of entities, and in English is often recognisable as a 'What is a ...?' question. This class is, in fact, appropriate to Lehnert's original domain as well as to an industrial problem-solving domain, and would have been worthy of consideration in her work.

There is considerable structure in the classification itself. Lehnert simply postulates her categories, leaving them organised only into a single level. Relationships exist between the categories, but these relationships are not made explicit. The next section clarifies and organises these relationships between categories, using the ideas underlying the description of primitive script-manipulation tasks, since in answering questions one must indeed undertake various movements around a causal chain. This will yield a more richly descriptive and deeper hierarchy into which the categories can be fitted.

6.2.3 The Overall Hierarchy of Question Categories

Instead of a shallow, flat tree of dependencies, the structure shown in Figure 44 may now be obtained. The classification follows the distinctions drawn between varieties of script-manipulative acts discussed in Chapter

Classifying questions in this way stems directly from the view of question-answering as the process of moving in characteristic ways around a causal chain. A primary

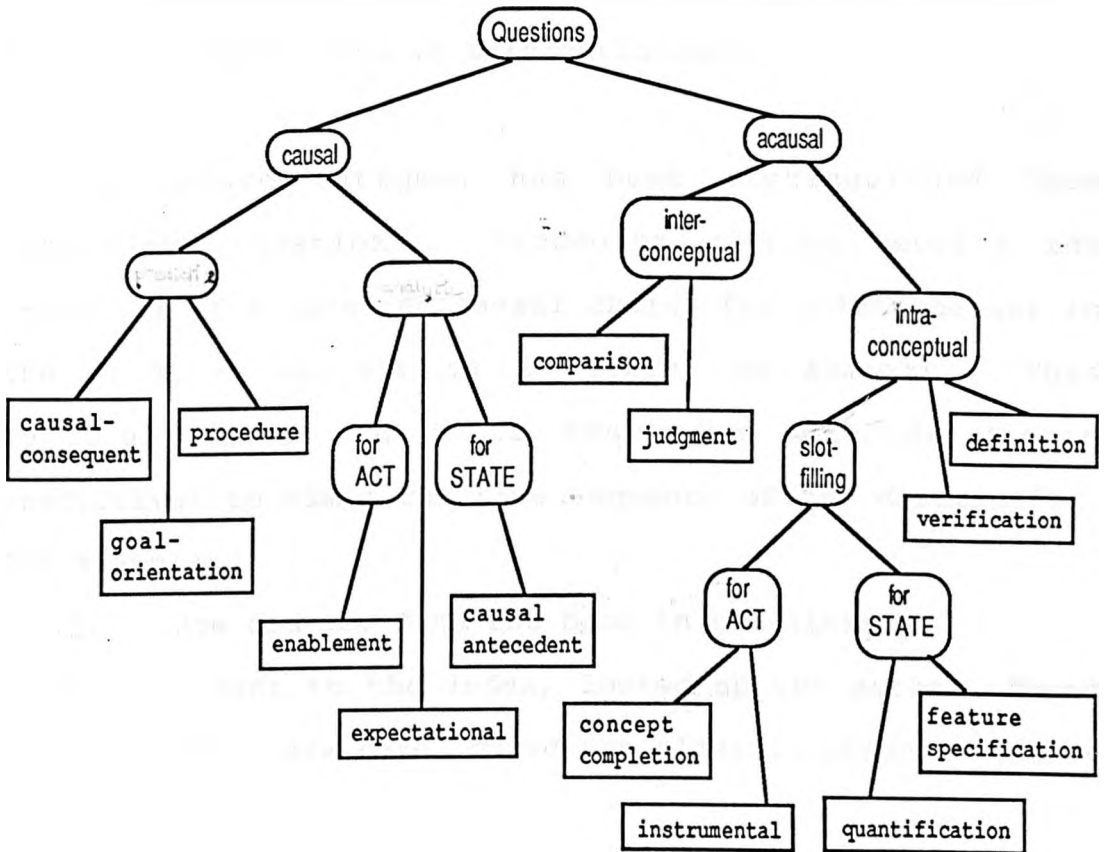


Figure 44

distinction is made between questions which require movement along causal links and those which do not (acausal). The causal category is then divided into analytic and predictive classes.

The predictive class is characterised by a movement of

attention from the question concept in the causal chain toward a resultant state or enabled act. It is this forward traversal of the causal links that is common to each of the three categories in this class. The identification of causal consequent and goal orientation as predictive categories is straightforward.

The procedure category has been distinguished from instrument questions. Procedure queries require the traversal of a detailed causal chain, for which the act in the question concept is a 'title' or summary. This detailed causal chain is traversed forwards (hence predictive) to mimic the time sequence of the description. For example:

- Q. How did you find the book in the library?
- A. I went to the index, looked up the author, found the index card, noted the classification number...

Categories in the analytic class are similarly related by the need to traverse causal links in finding an answer to the query. In this subclass, however, the causal links are traversed back up the causal chain; moving from the question concept toward enabling state or causative act. The distinction between causal antecedent and enablement has been more sharply defined. An enablement question has an act as a question concept, and enquires about the state which enabled it. Symmetrically a causal antecedent query asks about an act which caused the state in the question

concept. The expectational category may have either an act or a state as its question concept.

The acausal class is also divided in two. The intraconceptual subclass requires that only the image or entity in the knowledge base matching the question concept is examined. Relationships, causal or otherwise, with other images or entities are not relevant to the answering of the query. Several categories constitute one sub-class of the intraconceptual class. These are the slot-filling categories. They can be further divided into slot-filling for acts, slot-filling for states and slot-filling for individual entities. The instrument category is a slot-filler for an act. It is interesting that a shift in the level of detail required can transform the relatively simple instrument query into the more complex causal procedure question. For example:

Q. How do you get to London ? (instrumental)

A. By train.

Q. How do you get to London? (procedural)

A. You take a taxi to the railway station, buy a rail ticket, get on the train ...

Methods for disambiguating the natural language question must depend on the context in which the question is asked and on the responder's view of the enquirer, but this subject lies outside the work reported in this thesis.

The interconceptual subclass requires consideration of two entities, which are related not by causal links, but by relationships within some hierarchy. These relationships are often 'sibling' links, in that both entities are instances of the same domain class. For example, the two entities might be a chlorinated rubber coating and a drying-oil coating: both are instances of the 'coating' superclass.

The two categories in the interconceptual subclass, the judgment and comparison categories, are related to each other. Examples of the comparison question category are:

Which is easier to implement, causal consequent or goal orientation ?

Is drying-oil-paint more viscous than a chlorinated rubber coating ?

An example of the judgment question category is:

What do you think of bitumen coatings ?

The judgment class of query is seeking information about a comparison of the object of the question concept with some ideal in the same hierarchy; in the example, this is the coating hierarchy. This notion of comparison with an ideal can be seen most clearly if a comparison question is contrasted with a judgment query having a closely related question concept.

Q. Which is more resistant to mechanical damage, a chlorinated-rubber coating or a bitumen coating ?

A. A bitumen coating.

Q. What do you think of bitumen coatings ?

A. ...bitumen coatings have poor resistance to mechanical damage...

So the bitumen coating wins over the chlorinated-rubber coating, yet is judged 'poor' in relation to some other unspecified standard. This standard or ideal can be thought of as an aggregation of the strongest attribute values available in the relevant hierarchy. So if the best possible available mechanical damage resistance is obtained from sprayed-metal coatings, then this is the criterion against which all other coatings will be judged in a general sense; that is, as intended in a judgment question.

6.2.4 Analysis of HOW and WHY using the revised classification

The aim of this analysis is to distinguish

- i the nature of the question concepts
- ii the correct question category for HOW and WHY questions in rule-based systems.

6.2.4.1 The WHY question

Consider this example dialogue fragment from a rule-based system for selection of anti-corrosive paints.

What is the standard of surface preparation?

** WHY

I am trying to establish a value for COATING ADHESION

Rule: If the surface preparation > 2

Then coating adhesion is good

Consider the meaning of the question that is being answered here.

The question can be expanded in English in several ways.

Why did you ask me that question?

Why is that concept important?

Why do you need to know?

In what way will that information be used?

The question concept deals with the availability of information:

You asked me a question.

That concept is important.

You need to know.

That information will be used.

The contents of 'that question', 'that concept' or 'that information' are only of secondary importance. The central acts or states in the question concepts are concerned with the manipulation of information; 'ask', '...is important', 'need to know...' and 'information ... used'. All the expansions, in more or less circuitous ways, are enquiring about the system's current 'goal', that is, toward what desirable state it can move when the sought-after

information is provided. These are system-level acts and states, often taking whole domain-level assertions (i.e. 'that concept') as objects of the verbs. The most appropriate category for these WHY questions is goal orientation. Here, the system is functioning as the sentient agent which has definable goals. Its actions in pursuing those goals form the question concept of the WHY question.

6.2.4.2 The HOW question

The HOW question can be elaborated in a manner similar to that used for WHY.

What made you reach that conclusion?

(causal antecedent)

What enabled the system to prove that? (enablement)

Also similarly, the question concepts are at the system level.

You reached a conclusion.

The system proved something.

The questions are asking about the states of knowledge which have led to a certain conclusion being drawn. It is tempting to think of HOW questions as procedural or instrumental. If this were so, the following would be obtained:

Q. How did the system prove that?

A. By modus ponens

The enquirer does not want this description of a mechanism,

nor a description of any detailed instantiation of the mechanism. The useful answer is in terms of enabling states.

6.2.5 The Utility of the Question Categories

It is useful to employ the classification and the idea of a 'question concept' to delineate the extent of what Clancey calls the "aphasia" of single-level rule-based systems [Clancey 1983].

Kidd [Kidd 1985] gives general examples of questions commonly asked of human experts by callers in radio phone-in programmes. She points out that the user plays a major role in the problem-solving dialogue. The user has views on the essential components of the problem definition, and often on the nature of an acceptable solution. This leads to a negotiation between the expert and the user. Many of the questions put to the expert belong to categories other than those encompassed by HOW and WHY. For example:

Is X a good remedy ? (Judgment)

Which is the best remedy: X, Y or Z ? (Comparison)

How does remedy X work ? (Procedural)

Why doesn't remedy X work ? (Expectation)

What happens if remedy X is used ? (Causal Consequent)

It is also interesting to note that the question concepts concern domain-level entities. In reality, then, it seems

that the user asks questions in categories which span the whole hierarchy, examples being present for the analytic, predictive and acausal classes; and that these questions concern actions in the physical world rather than the information processing acts of the expert.

6.3 Question Answering in ADEPTUS

6.3.1 Introduction

In designing QUALM, Lehnert was dealing with only a single level script. In ADEPTUS, however, I have represented the system's knowledge on two levels, the d-script and the s-script. Questions can then be asked about processes and states occurring in either of these scripts. No level-specific constraints are in effect on a question category. Any question category which is valid for applied science problem-solving can be posed at either level. This does not mean, of course, that all question types would necessarily yield useful information at each level; only that there is no a priori reason why that query class may not be used. Syntactic restrictions, however, do apply on both levels of script. These follow Lehnert's rules. For example, a goal orientation question cannot sensibly be phrased to ask about processes in the physical domain, since by the definition of the term 'physical process' no sentient actor is available to have goals ascribed to them.

6.3.2 The Goal Orientation Category

6.3.2.1 Questions about the problem-solving process

As discussed above in section 6.2.4.1, this class encompasses the WHY questions familiar from rule-based systems. In a rule network, the goal orientation question is concerned with the problem-solving behaviour of the system, although to the naive user it can often seem as though the physical processes of the domain are the subject of the query. However, the lack of explicitly represented knowledge of physical causality in the domain leads to an inability to ask domain-level goal-orientation questions.

In the rule-based system, the answer to a goal orientation (WHY) question takes the form of the next conclusion obtainable from the firing of a single rule, where the premise of that rule contains a term matching the piece of information which prompted the question. The same question can be asked of ADEPTUS, but the answer is obtained in a rather different manner. ADEPTUS requires a description of the problem situation before any script instantiation or manipulation is undertaken. The user's goal-orientation query may be expressed while the **situation-description** schemata are being filled, before problem-solving begins; or it may be expressed when the selection script has been executed. In either case, the technique for answering the question remains the same. In the illustrative example

that follows, the uninstantiated script is employed. ADEPTUS is given two pieces of information; a question concept and the question category - in this case, goal orientation. The question concept for a goal orientation question must be an active concept.

For example:

Why does ADEPTUS need a value for the substrate
surface-key ?

Question concept:

(ACTOR ADEPTUS) (ACT place-value)
(OBJECT (substrate surface-key))

Question category:

Goal orientation

To answer a goal orientation question, we must establish first the state that results directly from the execution of the act described in the question concept. The subgoal requiring this act is then the execution of the next act or process that is directly enabled by the state achieved.

The state achieved by our example question concept is the existence of a value in the surface-key slot of the substrate schema. ADEPTUS performs a 'predictive scan' down the causal chain, looking for a state that has the existence of this value as an active enabling factor. Attention moves down from a state via the *enables* link to a process/act, thence to the next state via the *results-in* link. As each state in the script is encountered, ADEPTUS checks whether or not this stative image names the value of

the substrate schema's surface-key slot as an active enabler. (See Fig. 45).

```
{{$s-script
  state1 (?$D ?SOL ?SIT)
    [active-enablers ((environment-description is-a+inv)
                      (available-tool is-a+inv)
                      (substrate-description is-a+inv)) ]

  act1 (MERGE ?$D ?SIT)
    [affected-by ((environment-description is-a+inv)
                  (available-tool is-a+inv)
                  (substrate-description is-a+inv)) ]
  . . .
}}
```

Figure 45

The *is-a+inv* slot name is produced by the system in the absence of a named inverse for a slot, when the SRL inverse-creation switch is on. It represents the inverse of the *is-a* link.

Following the progress of the s-script, illustrated in Figure 9, it can be seen that the states enabling the MERGE and RUN acts do not require a value for any substantive domain-level schema slots. The two kinds of entities that they do require are instantiated d-script slots, and the existence of certain entire domain-level schemata. The existence of the latter is established by an examination of the appropriate class hierarchy. Consider for example the MERGE act and its enabling state. The execution of this

act requires values in the schema-slots named in the active-enabler facet of the enabling state. These are identical to those named in the affected-by facet of the MERGE act. This exact reciprocity is only the case when a state enables exactly one active image. (An example of an asymmetric situation is given in section 6.3.2.2)

The slots which must have values for this MERGE act to take place are those that show the names of child schemata of the **environment-description**, **available-tool** and **substrate-description** schemata. That is, the domain-level script cannot be MERGED with a problem situation description until instances exist of the three known components of the problem description. To illustrate the second possibility for s-script act requirements, consider the RUN act and its enabling state, shown in Figure 46.

```

{{ $s-script
  . . .
  state3 (? $D)
    [active-enablers (( $d-script state1)
                     ( $d-script process1)) ]
  act3 (RUN ? $D)
    [affected-by (( $d-script state1)
                 ( $d-script process1)) ]
  . . .
}}

```

Figure 46

The slots actively enabling the RUN act are the initial

slots of the d-script. If an instantiation is available for the two named slots, then the RUN act can execute. If the state being examined in the predictive scan does not have the schema-slot value of interest as an active-enabler, ADEPTUS passes over that state, unless the enabled act is itself a predictive task. When a predictive task is encountered, the attention of the predictive scan shifts to the script that is the object of the new task, and ADEPTUS continues the predictive scan down this 'object' script in the same manner as before.

When a state is found that uses the value of the schema-slot combination specified in the question concept to enable the ensuing process/act, the information required to construct the answer can then be accessed.

The actor whose subgoal is being established is still ADEPTUS. To answer the question, we must now determine what the actor is attempting to do at the target point. ADEPTUS is attempting to predict the outcome of the (STICK ?C ?S) process, and that process cannot be executed until the **substrate surface-key** value is known. It is within the stative image of the **\$d-script state2*** value that we regard the target schema-slot combination as being actively matched. That is to say, although a value for that slot may have been available prior to that state, *state2* of the **\$d-script** schema is the first instance we have detected of its active enabling influence on a process/act. The

* See Fig. 46a, page 189

movement down the *enables* link from the match to the enabled process is the predictive task that characterises the goal orientation answering mechanism and assigns the goal orientation category to the causal/predictive subclass of query types. The hypothetical RUN act that ADEPTUS is attempting to execute when the **substrate surface-key** becomes an enabling factor is composed of repeated calls to a primitive PREDICT act, which is actually responsible for the execution of the individual processes/acts in the script being RUN. This allows us to construct the essence of the answer to the query thus:

```
(ACTOR <ACTOR of question concept>
  has-goal ( <ACT or PROCESS description> )
```

The actor of the question concept is ADEPTUS. The active image identified as the goal is:

```
(ACTOR ADEPTUS) (ACT PREDICT)
(OBJECT
  ((STICK ?C ?S) affected-by (substrate surface-key)))
```

So the complete value is:

```
((ACTOR ADEPTUS) has-goal
  ((ACTOR ADEPTUS) (ACT PREDICT)
  (OBJECT
    ((STICK ?C ?S) affected-by (substrate surface-key))))))
```

In English, the question is

Why does ADEPTUS need a value for the substrate surface-key?

The answer obtained indicates that

ADEPTUS is trying to PREDICT the outcome of the surface

and the coating STICKing together, and this is affected by the surface-key of the substrate.

Is this a reasonable answer to the question? I would argue that it is. The level of the question and answer are the same; both are centrally concerned with problem-solving images rather than domain-level ones. The answer does indeed tell the user something interesting about the motivation for requiring the substrate surface-key information. It does NOT, however, give any details about the way in which the substrate surface-key affects the STICKing process. This is not surprising. The effect of the substrate surface-key value differs for each coating about which the system knows, i.e. differs for each solution known to the system. To ask how the substrate surface-key affects a particular coating is a domain-level concept-completion question.

Lehnert describes Goal Orientation questions as those that involve a REASON link. The answer is found by identifying the unknown image which is the REASON for the question concept. The need for such a link seems to be the complexity and deviousness of human motivation in the 'ordinary occurrences' domain. ADEPTUS does not require REASON links; all of its motives are explicit, and are directly led to by the achievement of the constituent subgoals.

6.3.2.2 Goal Orientation questions about domain-level acts

Goal orientation questions can, of course, be asked about acts carried out in the physical domain modelled in the system. To illustrate such questions, I shall employ a piece of causal chain that does not appear as part of the current version of ADEPTUS. This is because no acts (as distinct from processes) occur in the d-script, for reasons of simplification. Again, no REASON links are required; I assume the motives of the people involved to be straightforward. This clearly renders the system incapable of understanding and hence discussing motives such as laziness, greed, sabotage etc. Consider the piece of causal chain illustrated in Figure 47. This describes the preparation of the substrate, prior to paint application. This script fragment is shown fitted on to the front of the existing d-script, replacing the TOUCH process and its enabling state.

I shall consider the Goal Orientation question

Why does someone abrade the substrate ?

Question Concept:

((ACTOR UNKNOWN) (ACT ABRAD) (OBJECT ?S))

Question Category:

Goal Orientation

{{ \$d-script

```
stateA (?OPERATIVE ?S ?TOOLS ?LIQUID)
  [instance stateslot ]
  [enables actA actB ]
  [active-enablers ((cleaning-tool tool-used)
                    (operative skill-level)
                    (dip-liquid constituents)
                    (dip-liquid temperature)
                    (dip-liquid pH)) ]
actA ((ACTOR ?OPERATIVE) (ACT DEGREASE) (OBJECT ?S)
      (INSTR ?LIQUID))
  [instance actslot ]
  [results-in stateC]
  [affected-by ((dip-liquid constituents)
               (dip-liquid temperature),
               (dip-liquid pH)) ]
  [can-affect ((substrate constituents)
              (substrate surface-key)) ]
actB ((ACTOR ?OPERATIVE) (ACT ABRABE) (OBJECT ?S)
      (INSTR ?TOOLS))
  [instance actslot]
  [results-in stateC]
  [affected-by ((cleaning-tool tool-used)
               (operative skill-level))]
  [can-affect ((substrate surface-key)) ]
stateC (?PAINTER ?S ?C ?TOOLS)
  [enables processC]
  [active-enablers ((painter skill-level))]
processC ((ACTOR ?PAINTER) (ACT APPLY) (OBJECT ?C) (TO ?S)
         (INSTR ?TOOLS))
  [results-in state2]
  [affected-by ((painter skill-level)) ]
  [can-affect ((coating constituents)) ]
state2 (PHYSCONT ?C ?S)
  [instance stateslot ] . . . ]}
```

Figure 47

This is easily identifiable as a domain-level question since the ACT involved is a domain-level ACT. This indicates that the relevant script is the d-script. Following the format established for the system-level goal orientation questions, a predictive scan thus commences at the start of the d-script. The question concept is matched by the value of `substrate`. Once more, to answer a goal orientation question, we must

- (a) establish what schema-slot combination is affected by the execution of the question concept ACT
- (b) continue with the predictive scan until a subsequent ACT or PROCESS is found that is actively enabled by the target schema-slot combination.

The ACT of abrasion, as we see from Fig. 47, can-affect just one schema-slot combination, the **substrate** surface-key. The predictive scan therefore continues down the d-script, looking for a state which names the substrate surface-key as an active enabler. Of course, since this is the d-script, none of the processes/acts can themselves be predictive tasks, so we cannot drop down into a lower-level script. An active match for the target schema-slot combination is found in the state enabling the STICK process. Once again it is possible to formulate an answer to the question, using the template

```
(ACTOR <ACTOR of question concept>
  has-goal ( <ACT or PROCESS description> )
```

We instantiate the ACTOR using the question concept ACTOR, and the active image with the available details of the STICK process. So the complete answer to the question is:

```
((ACTOR UNKNOWN) has-goal  
  ((STICK ?COATING ?SUBSTRATE) affected-by  
    (substrate surface-key)) )
```

In natural language, the gist of this is that someone is trying to make the coating stick to the substrate, and this is affected by the surface-key of the substrate. The major difference between this answer and the example at the selection-script level is the absence of the PREDICT act. Here, we are discussing a person operating at the level of physical processes; in the previous section's example, the discussion was of the doings and goals of the system.

In both the system-level and the domain-level examples given, the target schema-slot combination is identified in a purely linear section of the script. That is, the state in which the active match occurs enables only one active image. However, if the state enables more than one act or process, then the symmetry between the active-enablers facet in the state and the affected-by facet in the active image no longer holds. In order to find the relevant enabled act, each of the enables links must be pursued and the value of the affected-by facet for each active image examined. For example, consider the question

Why does the surface-preparation operative undergo a training period ?

The act of undergoing a training period will change the operative skill-level value. This state attribute is actively matched in *stateA* of the **\$d-script** schema illustrated in Figure 47. *stateA* enables both **actA** and **actB**; to find the active image required as a response, we must follow both of the *stateA* enables... links. The first of these leads to **actA**. On examination of **actA**'s affected-by facet, we see that this act is not affected-by the target schema-slot combination of **operative skill-level**. The next enables link is then followed to **actB**, where the affected-by facet contains the target combination. It is this act which is then used to construct the answer to the question.

6.3.3 The Judgment Category

6.3.3.1 Introduction

Judgment questions form a category in the interconceptual subclass. They are given the name of a concept as their question concept and are required to express an 'opinion' about that concept. For example, the question may be

What is your opinion of a drying oil paint in this situation?

This question is clearly being asked after a problem situation has been described. Information about the problem situation is essential if the system is to be able to answer such a question: without a problem context, it is difficult to pass judgment on the coating. When a human is asked the general Judgment question

What do you think of drying oil paints ?

he or she has two options. One is to ask for more constraints on the question by requesting a problem context in which to frame the reply, ie by responding with another question:

In what context?

Or, the human can respond by identifying contexts in which high-level judgments can be made on the question concept.

For example:

Drying oil paints are good when the surface preparation is poor, but are susceptible to spillages of dilute chemicals.

This style of response is not comparing the coating with some standard, but is attempting to establish general characteristics of the coating. The answer given can be seen as the answer to two quite different questions:

Which adverse situation factors still give good results for drying oil paints?

Which average situation factors give poor results for drying oil paints?

It is interesting to note that the answer is phrased as a

collection of heuristics about drying oil paints.

These are analytic questions, although the original natural language query seems to imply a judgment. The problem of identifying the correct question category from the natural language form of the question has already been achieved by Lehnert, and so is not addressed in this thesis.

6.3.3.2 Answering Judgment Questions

Judgment questions as illustrated by the question

What do you think of drying oil paints in this situation? are answered in ADEPTUS by a comparison with an idealised concept. The act which does this is already defined as part of the s-script and is called COMPARE. If a problem situation has been defined (i.e. enough is known about the physical environment and surface preparation to allow a prediction of the worn dry film to be made), the comparison is straightforward. The COMPARE act takes two concepts, the central question concept and its corresponding idealised form, and from these produces a Divergence schema (see section 5.3.3.5). The Divergence schema is used as the basis for constructing the reply, but itself contains information which is not used.

A Judgment question is interested in two groups of attributes in the Divergence schema: the particularly good and the particularly bad. These are the slot values which

will give an interesting response. For example, if the COMPARE act produces the Divergence schema Divergence6 (see Figure 48), only the **value-judgment** type of slots which have 'very' or 'extremely' as qualifiers will be mentioned in response to the Judgment question.

```
{{Divergence6
  is-a Divergence
  between (dry-film-drying-oil-paint2 Ideal)
  adhesion (very poor)
  state (average)
  defects (fairly good)
  hardness (fairly good)
  flexibility (fairly poor)
  finish (extremely good)
  thickness (extremely good)      }}
```

Figure 48

So, the question is:

Question concept :

dry-film-drying-oil-paint2

Question category :

Judgment

And the answer is constructed from the Divergence6 schema.

```
dry-film-drying-oil-paint2
and   has extremely good finish
      has extremely good thickness
but   has very poor adhesion
```

The general form of the answer is :

<Question Concept>

has <slot value> <slotname>

and <slot value> <slotname>

...

but has <slot value> <slotname>

and <slot value> <slotname>

...

The actual values of the **value-judgment** dimension are stored in numeric form, from -3 to +3. It is easy, therefore, to distinguish between positive and negative value judgments.

When a Judgment question is asked about a concept for which a Divergence schema already exists (i.e. one which has previously been considered in the problem-solving task) it could be argued that the apparent Judgment classification of the query should be some form of slot-filling. This ignores the essential nature of the judgment category. The COMPARE act must happen at some point for such a question to be answered: it is fundamentally an interconceptual task.

6.3.4 The Expectation Question Category

Expectation questions are characterised as 'Why not?' questions. They form one of the analytic categories and

enquire about the causes of a state or act not occurring. This enquiry about past causes is what leads to the categorisation of expectational questions as analytic.

The question concept may be either active or stative.

Why isn't the pavement wet? (Stative question concept)

..because it didn't rain.

Why didn't it rain? (Active question concept)

..because there were no clouds.

An expectational question can be asked either at the physical level, i.e. about a process or state within the d-script, or at the problem-solving level, concerning an act or state in the s-script.

If the question concept is a state, then the answer is obtained by examining the act or process resulting in that state. For active question concepts, the answer lies in the state which enables the act. For example, consider the question:

Why doesn't the coating stick to the substrate?

The question concept is active:

The coating sticks to the substrate.

In the d-script, this matches the value of the *process2* slot,

(STICK ?C ?S)

The enablement of this process is governed by the details of *state2*,

(PHYSCONT ?C ?S)

so to answer the question it is necessary to look at the binding-list facet for *state2* to establish what aspect of the state is disabling the process.

Let us say that the *constituents* slot of the **substrate** schema has the value

((grease 70)(steel 30))

that is, a film of grease covers 70% of the steel substrate. The rulebase which defines the macro-effect of the STICK process uses the value of this slot to decide on the eventual relationship between the coating and the steel substrate. In the example above, the presence of a grease deposit on the substrate changes the resulting state value from

(PHYSCONT ?C ?S)

to (?C ?S)

The identification of expectational questions as analytic tasks can be seen in the path taken from the point in the script at which the question concept is matched, to the information which determines the disabling of the question concept. The premises of the rules in the STICK *macro-effect* rulebase require that the binding for the substrate in the enabling state be obtained. Central to this is the requirement to follow the enabled-by link back up the causal chain structure. This is the characteristic movement which identifies cause-seeking tasks.

The rulebase describing the macro-effects of the STICK process contains the rule

```
IF ( percentage grease substrate-constituents ) > 5
THEN replace resultant-state (PHYSCONT ?C ?S) (?C ?S)
```

[where:

substrate-constituents is an association list obtained from a schema named in the binding list of the previous state

percentage is a function which returns the value associated with a nominated key on an a-list

resultant-state is the location of the value of the state resulting from the process]

Then the question

Why didn't **coating1** stick to **substrate1**?

Question concept:

```
( STICK coating1 substrate1 )
```

Question category:

Expectation

can be answered:

```
( STICK coating1 substrate1 )
```

did not happen successfully because

```
percentage of grease in substrate-constituents > 5
```

This category has not yet been implemented. The description is included to illustrate methods of response to cause-seeking questions.

6.3.5 The Definition Question Category

Definition questions are characterised in English as

'What is a...?'

questions. The question concept is neither a state nor an act, but is a single entity. This entity may be used as part of a state description or to participate in an act or process, but it is not sufficient to describe a full image.

An example Definition question is:

What is bitumen ?

Answers are obtained by simply using the information present in the schema of that name. If a slot value is present, or is obtainable by inheritance, then the slot name and its value are both given. If the slot is present (or can be inherited) but has no value, then the default is given if available, with a qualifier indicating that it is a default value. Otherwise, the range facet can be examined to provide information about constraints on the value of a slot. As a last resort, if even range information is lacking, it may simply be stated that the concept can have a characteristic having the name of the slot.

Question concept

Bitumen

Question category

Definition

Answer:

Bitumen

is-a **coating**

with *colour* default **black**

with *constituents* **natural-hydrocarbons**

with *state* one-of (**solid liquid**)

with *adhesion* range ((extremely weak) (extremely strong))

with *single-film-thickness* range (200 500)

with *defects* one-or-more-of (**pinholes blisters**)

with *hardness* range ((extremely soft) (very hard))

with *flexibility* default (fairly flexible)

with *finish* **dull**

The slot names and corresponding values in a frame form the essence of the definition of a named entity.

6.4 Summary

The three question types which have a preliminary implementation in ADEPTUS are goal orientation, definition and judgment. A fourth category of expectation questions has been discussed to provide illustrations of question types from each of the major branches of the question classification tree. Images from both the s-script and the d-script are susceptible to use as question concepts; the essential ideas of characteristic movement around a causal

chain are unchanged whatever the level.

Further examples, illustrating ADEPTUS' response to questions posed by Unilever personnel, are given in Appendix II.

```
{{ $d-script
  state1  (?EA ?C ?S) . . .
  process1 (TOUCH ?C ?S) . . .
  state2  (PHYSCONT ?C ?S)
    [active-enablers ((substrate-description constituents)
                     (substrate-description surface-key)) ]
    [active-results ((coatings single-film-thickness)
                    (coatings defects)
                    (coatings time-between-coats)) ]
    [enables        process2 ]
    [produced-by    process1 ]
  process2 (STICK ?C ?S)
    [affected-by ((substrate-description constituents)
                 (substrate-description surface-key)) ]
    [can-affect    ((coatings adhesion)
                   (coatings
viscosity)) ]
    [prerequisites state2 ]
    [results-in    state3 ]
  state3 (?ED (PHYSCONT ?C ?S)) . . . }}
```

Figure 46a

7 CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

7.1 Conclusions

7.1.1 Dialogue with Expert Systems

The aim of the work described in this thesis has been to widen the understanding of explanation in problem-solving expert systems. No satisfactory theory of explanation is available at present, and until this is remedied, a good approach to understanding the nature of explanation is to investigate the mechanism by which humans seek it. Questions are the most direct and overt form of explanation-seeking. In understanding the questions that can be asked in a domain, and examining the questions that actually are asked in natural dialogues, we are beginning to make progress in designing problem-solving representations that are capable of answering such questions when they are overtly asked.

In applying the ideas of natural language question answering to expert systems, this thesis delineates just how poor the explanation provided by the currently accepted HOW and WHY questions really is. It provides a greater insight into a problem that has been recognised by many 'knowledge engineers', but helps to move the difficulty from an ill-defined sense of insufficiency to a set of

specific shortcomings in the form of whole categories and levels of questions which a conventional expert system is fundamentally incapable of answering.

An interesting point that has emerged in the thesis is the value which can be obtained from the use of existing work in the natural language understanding area for systems that have a technical domain. Now that Schank and other workers in natural language processing are beginning work on technical domains instead of social ones, we can anticipate further significant advances of benefit to knowledge-based systems.

Simple causal chains offer great promise for representation of physical processes. Their simultaneous use for describing the system's problem-solving processes offers a uniformity of control structure, yet the distinction between problem-solving and domain levels remains unambiguous because of the different acts or processes characteristic of each level.

Scripts are not intended to function as the major problem-solving mechanism but rather as a mechanism for communication. When two or three good candidate solutions have already been identified by a conventional system, scripts offer a particularly useful method of communicating more detailed knowledge about each solution to the user. This is itself a form of improved explanation, of

communicating the system's understanding of the situation to the user. The same representation has been shown to support several types of question answering, demonstrating the potential of the description to permit free access to the knowledge contained in the system.

The categorisation of script-manipulative tasks, and hence the detailed categorisation of question types, is completely new. Understanding the nature of explanation is gravely hampered by the ad hoc collection of 'types of explanation' with no attempt to identify the relationships between the proposed classes. Using the classification, it will be possible not only to build real systems having far greater expressive power, but also to begin to quantify the communicative shortcomings of existing systems.

The implementation of the ADEPTUS system is preliminary, and to some extent fragmentary. SRL, as available at Liverpool University, has serious shortcomings and has no technical support. Only a sample of the question categories identified have been implemented, and there is no reason to believe that any of the remaining categories would pose substantial problems. The Request category is an exception and presents peculiar problems. These problems are, however, ones of identification; and as such identification is the responsibility of the natural language interface, they have been omitted from the system.

Although the system is incomplete, enough has been achieved however to indicate that the methods suggested by this thesis are significant and offer a means to improve the explanatory capabilities of expert systems.

7.1.2 The Relationship between Problem-Solving and Question-Answering

I have used question answering as a means of approaching the ill-defined problem of 'explanation', and have distinguished between the problem-solving activities of the system and its explanatory capabilities in the form of the answering of explicit questions. This dichotomy is more apparent than real. In dialogues between enquirers and human experts, the user usually initiates the problem-solving activity in the expert by means of asking one or more questions. Each of the existing expert systems performs a problem-solving activity which has an implicit question associated with it. For example, the questions that MYCIN can answer are

'What disease can produce these symptoms?'

'What therapy best treats the disease in this case?'

XCON answers the question

'How can this order be configured satisfactorily?'

The processing necessary to answer these questions is represented in the control structure of the appropriate

systems. The automatic search for an answer to the system's 'permanent' question can sometimes cause problems: in my own experience I have seen people obtain misleading results from a simple rule-based paint selection system because they did not fully understand the system's question. Several users assumed that the 'problem' was simply

'What paint is best for this situation?'

Whereas the implicit question from the user was actually

'What paint is best for a steel substrate in this situation?'

While posing the problem overtly in the form of a question does not automatically remove this type of error, systems that expect an overt problem statement would need to be inherently more flexible and to make fewer assumptions. A further guard is of course that a system flexible enough to solve several types of problem in the same domain will necessarily have a powerful question-answering capability. The user thus has the opportunity to explore any solutions or misunderstandings, and hence a greater opportunity of correcting misapprehensions.

So, what current expert systems are doing (ignoring HOW and WHY facilities) is answering a single question. I see no reason to believe that these questions cannot be analysed and categorised in the way discussed in this thesis. The problem-solving, which we have so far believed to be the 'real' expertise, is in fact only a subset of the wider

field of question-answering by knowledge-based systems.

'Explanation' (like 'intelligence'), is a protean word. A good explanation changes with the explainer, with the listener, with the subject matter, the purpose, and the medium of the explanation. The idea of explanation as a single activity is a myth, and we must begin to deal with the problem in the most practical but flexible way we can. Answering the widest possible variety of questions is a good place to start.

7.1.3 Heuristics and Levels of Knowledge

I have used two levels of script, the physical model and the 'problem-solving' representation. These levels are, in a modified form, those proposed by Clancey [Clancey 1983]. From Patil's work [Patil 1981], it is clear that the domain model itself has different levels, and this is reflected to a small extent in my use of the demons' rulebases to establish a value for a slot in the **\$d-script** schema. If more detailed knowledge were available in the domain, these rulebases could themselves be represented as scripts, dealing for example with microscopic chemical processes. Applied science domains, however, always seem to reduce to a collection of heuristics: even the experts rarely know the precise nature of causality at very detailed levels. Whether for theoretical or practical reasons, then, it

seems inevitable at our present state of scientific knowledge that the ultimate means of predicting the most detailed results will remain heuristic. At such levels, our forms of explanation are forced to remain those of the familiar HOW and WHY questions. This reflects the dependence of explanation on understanding. At the most detailed levels of knowledge, heuristics represent a failure to understand a mechanism, but are a way of obtaining useful information despite this. However, while it is still possible to elicit a causal explanation from an expert, it seems a worthwhile venture to attempt to build systems that also have this capability.

7.2 Directions for Future Research

7.2.1 Question Categories

Further work is required on the completeness of the question classification presented here. Transcripts of series of dialogues between enquirer and expert are needed to resolve at least three obvious points of interest. First, what categories of questions exist which are not described in the existing classification? And can these new categories be sensibly assigned a place in a modification of the hierarchy?

Next, the possibility that question categories are

dependant on the domain of expertise must be investigated. For example, Lehnert's request category is redundant in an expert system. This redundancy stems more from the medium of communication than from the domain; if an enquirer approaches a human expert, the request category may still be used, whereas communication with a machine does not require such 'questions'. The possibility that question categories are domain-dependant remains unconfirmed, but comparisons of dialogue analyses from different domains will clarify this.

Third, a very practical problem is the extent to which expert systems in different domains should attempt to incorporate the many established question categories. Little is as yet known about what users really ask. Again, dialogue analyses for a particular domain will give some indication of the commonest or most crucial question types used by the enquirer. Armed with this kind of information, the knowledge engineer can make an informed decision on the question categories that should be implemented as a priority to optimise the user's communication with the system.

7.2.2 Domain Acts and Processes

The processes used in the d-script in ADEPTUS are candidates for the term 'primitive', and suffice for the

purposes of the system. However, a more thorough investigation of applied science domains is necessary to produce a useful set of primitives for practical domains. Within the same domain there is also an evident need for primitives at different levels of detail. For example, a description of the molecular processes occurring when a paint dries will require a rather different vocabulary of process primitives than is needed when discussing the engineer's view of the event. In ADEPTUS' terms, such vocabularies for different levels will enable the expansion of the rulebases describing the macroscopic effect of domain processes. The rulebases could then be described as causal chains, on a more detailed level, using the appropriate vocabulary. This will facilitate the answering of procedural questions; i.e. those which enquire about methods for the execution of a named process.

7.2.3 Implicit Questions

The work described in this thesis has been directed to the answering of explicit questions. In real situations with a human expert, it has been observed that the expert often offers explanation without an overt request from the enquirer. [Pollack et al 1983, Kidd 1985] This spontaneous provision of knowledge can be profitably viewed as providing answers to implicit questions from the user. For a system to be able to do this in an intelligent,

appropriate way, it will be necessary for the system to contain a representation of the cognitive state of the user. If the system can identify specific gaps in the user's knowledge of the domain, it may be feasible to identify questions which describe the missing knowledge, and to provide explanation by answering these system-constructed questions. Progress in such an area depends strongly on the development of good user models whose representation is compatible with the causal chain approach used in this work.

7.3 Summary

Explanation is a difficult and complex field, and like all such fields offers a rich choice of avenues for exploration. It is something that we as humans do constantly; when one listens for it, it is quite surprising how common an event it is. As an intelligent function that we wish to understand and mimic, it touches on many existing branches of Artificial Intelligence and Knowledge-Based Systems work. The communication of understanding is fundamental to the design of systems which give their users good quality advice, and despite the difficulty of the task we must move towards giving explanation its correct place, in centre stage.

APPENDIX I

BIBLIOGRAPHY

- ADDANKI S. and DAVIS E.
'A Representation for Complex Physical Domains'
Proc. Int. Joint Conference on Artificial Intelligence-9,
1985.
- AIKINS J. S.
'Prototypical Knowledge for Expert Systems'
Artificial Intelligence Vol.20(1983), pp 163-210.
- CHANDRASEKARAN B. and MITTAL S.
'Deep versus Compiled Knowledge Approaches to Diagnostic
Problem Solving'
Int. Journal of Man-Machine Studies Vol.19(1983),
pp 425-436.
- CHARNIAK E.
'On the Use of Framed Knowledge in Language Comprehension'
Artificial Intelligence Vol.11(1978), pp 225-265.
- CHARNIAK E.
'A Common Representation for Problem-Solving and Language
Comprehension Information'
Artificial Intelligence Vol.16(1981), pp 225-255.
- CLANCEY W. J.
'An Antibiotic Therapy Selector which provides for
Explanations'
Proc. Int. Joint Conference on Artificial Intelligence-5,
1977.

CLANCEY W. J.

'Tutoring Rules for Guiding a Case Method Dialogue'
Int. Journal of Man-Machine Studies Vol.11(1979), pp 25-49.

CLANCEY W. J.

'Methodology for Building an Intelligent Tutoring System'
Report No. STAN-CS-81-894, HPP-81-18, Stanford University,
Stanford, Ca. 1981.

CLANCEY W. J.

'The Epistemology of a Rule-Based Expert System - A
Framework for Explanation'
Artificial Intelligence Vol.20(1983), pp 215-251.

CLANCEY W. J. and LETSINGER R.

'NEOMYCIN: Reconfiguring a Rule-based Expert System for
Application to Teaching'
Proc. Int. Joint Conference on Artificial Intelligence-7,
1981.

COOMBS M.J. and ALTY J.L.

'General Proposals for the design of a Knowledge-based
Consultant: Rationale and a Cartoon'
Department of Computer Science Report,
University of Strathclyde 1982.

COOMBS M.J. and ALTY J.L.

'Expert Systems: An Alternative Paradigm'
Int. Journal of Man-Machine Studies Vol.20(1984), pp 21-43.

* See note

CRAIK K.

'The Nature of Explanation'
Cambridge University Press, Cambridge, 1943.

CULLINGFORD R.E., KRUEGER M.W., SELFRIDGE M. & BIENKOWSKI

M.A. 'Towards Automating Explanations'
Proc. Int. Jt Conference on Artificial Intelligence-7,1981.

* COOMBS M.J. and HUGHES S.

'Extending the Range of Expert Systems: Principles for the Design of a Computer Consultant'
Proc. Expert Systems 1982, Brunel.

DAVIS R.

'Interactive Transfer of Expertise: Acquisition of New
Inference Rules'

Proc. Int. Joint Conference on Artificial Intelligence-5,
1977.

DAVIS R. [1980a]

'Meta-Rules: Reasoning about Control'

Artificial Intelligence Vol.15(1980), pp 179-222.

DAVIS R. [1980b]

'Content Reference: Reasoning about Rules'

Artificial Intelligence Vol.15(1980), pp 223-239.

DAVIS R.

'Reasoning from First Principles in Electronic
Troubleshooting'

Int. Journal of Man-Machine Studies Vol.19(1983),
pp 403-423.

DAVIS R. and KING J.

'An Overview of Production Systems'

Machine Intelligence Vol.8 (1977),
eds. E.W. Elcock and D. Michie, Ellis Horwood.

DAVIS R., BUCHANAN B.G. and SHORTLIFFE E.

'Production Rules as a Representation for a Knowledge-Based
Consultation Program'

Artificial Intelligence Vol.8(1977), pp 15-45.

DAVIS R., SHROBE H., HAMSCHER W., WIECKERT K., SHIRLEY M.,
and POLIT S.

'Diagnosis based on Description of Structure and Function.'
Proc. AAAI Conference, 1982.

EDMONDS E.

'The Man-Computer Interface: a Note on Concepts and Design'
Int. Journal of Man-Machine Studies Vol.16(1982),
pp 231-236.

FAGAN L.M., KUNZ J.C., FEIGENBAUM E.A., and OSBORN J.J.
'Representation of Dynamic Clinical Knowledge: Measurement
Interpretation in the Intensive Care Unit'
Proc. Int. Joint Conference on Artificial Intelligence-6,
1979.

FERRAND P.

'SESAM: an Explanatory Medical Aid System'
Proc. ECCAI 1984, Pisa.

FROST R. A.

'Identification of Similarities between Various Knowledge
Representation Formalisms'
Proc. Architecture of Large Knowledge-Based Systems
Conference, Manchester, 1984.

GOGUEN J.A., WEINER J.L. and LINDE C.

'Reasoning and Natural Explanation'
Int. Journal of Man-Machine Studies Vol.19(1983),
pp 521-559.

GERRING P.E., SHORTLIFFE E.H. and VAN MELLE W.

'The Interviewer/Reasoner Model: An Approach to Improving
System Responsiveness in Interactive A.I. Systems'
The A.I. Magazine Vol.3 No.4(1982), pp 24-27.

HAGERT G.

'What's in a Mental Model? On Conceptual Models in
Reasoning with Spatial Descriptions'
Proc. Int. Joint Conference on Artificial Intelligence-9,
1985.

HASLING D.W., CLANCEY W.J. and RENNELS G.

'Strategic Explanations for a Diagnostic Consultation System'

Int. Journal of Man-Machine Studies Vol.20(1984),

pp 3-19.

* See note

HUET G.

'In Defense of Programming Languages Design'

Proc. ECCAI 1982, Orsay, France.

HUME D.

'A Treatise of Human Nature'

Book I, Part III, Section XIV, 1738.

JACKSON P. and LEFEVRE P.

'On the Application of Rule-Based Techniques to the Design of Advice-Giving Systems'

Int. Journal of Man-Machine Studies Vol.20(1984), pp 63-86.

JOHNSON L.

'The Need for Competence Models in the Design of Expert Consultant Systems'

Int. J. Systems Research and Info. Science Vol.1(1985),

pp 23-36.

JOHNSON-LAIRD P. N.

'Mental Models'

Cambridge University Press, 1983.

KAHN G.

'On When Diagnostic Systems Want To Do Without Causal Knowledge'

Proc. ECCAI 1984, Pisa.

* HUGHES S.

'The Communicative Power of Expert Systems'

R&D Management Vol. 15 No.2 (1985) pp. 119-124

KIDD A. L.

'What Do Users Ask? - Some Thoughts on Diagnostic Advice'
Expert Systems 85, ed. M. Merry, BCS Workshop Series,
Cambridge University Press, 1985.

KIDD A.L. and COOPER M.B.

'Man-Machine Interface Issues in the Construction and Use
of an Expert System'
Int. Journal of Man-Machine Studies Vol.22(1985),
pp 91-102.

KOTON P.A.

'Empirical and Model-Based Reasoning in Expert Systems'
Proc. Int. Joint Conference on Artificial Intelligence-9,
1985.

LANGLOTZ C.P. and SHORTLIFFE E.H.

'Adapting a Consultation System to Critique User Plans'
Int. Journal of Man-Machine Studies Vol.19 (1983),
pp 479-496.

LEHNERT W. G.

'The Process of Question Answering'
Lawrence Earlbaum Associates, 1977.

McDERMOTT J.

'R1: A Rule-Based Configurer of Computer Systems'
Report CMU-CS-80-119, Department of Computer Science,
Carnegie-Mellon University, 1980.

MILLER P. L.

'A Critiquing Approach to Expert Computer Advice:
ATTENDING'
Pitman Advanced Publishing Program, Boston, 1984.

MINSKY M.

'The Society Theory'

in "Artificial Intelligence: An M.I.T. Perspective" Vol 1.
eds. Winston P.H. and Brown R.H. 1979

NECHES R., SWARTOUT W.R. and MOORE J.

'Explainable (and Maintainable) Expert Systems'

Proc. Int. Joint Conference on Artificial Intelligence-9,
1985.

PATIL R. S.

'Causal Representation of Patient Illness for Electrolyte
and Acid-Base Diagnosis'

Ph.D. Thesis, Laboratory for Computer Science, M.I.T.
1981.

POLLACK M.E., HIRSCHBERG J. and WEBBER B.

'User Participation in the Reasoning Processes of Expert
Systems'

Report MS CIS-82-9, Dept of Computer & Information Science,
University of Pennsylvania, 1982.

RIESBECK C.K.

'Knowledge Reorganisation and Reasoning Style'

Int. Journal of Man-Machine Studies Vol.20 (1984),
pp 45-61.

SCHANK R. C.

'Conceptual Information Processing'

North-Holland 1975. (a)

SCHANK R. C.

'The Structure of Episodes in Memory'

in "Representation and Understanding: Studies in Cognitive
Science" eds. D.G. Bobrow & T. Collins
Academic Press, 1975. (b)

SCHANK R. C.

'Representing Meaning: An A.I. Perspective'

Yale University Cognitive Science Technical Report 11,
1981.

SCHANK R.C. and ABELSON R.P.

'Scripts, Plans, Goals and Understanding'

Lawrence Earlbaum Associates, 1977.

SCHANK R.C. and CARBONELL, J.G.

'Re: The Gettysburg Address - Representing Social and
Political Acts'

in "Associative Networks" ed. N.V. Findler
Academic Press, 1979.

SCOTT A.C., DAVIS R., CLANCEY W. and SHORTLIFFE E.H.

'Explanation Capabilities of Production-Based Consultation
Systems'

Memo HPP-77-1, Heuristic Programming Project,
Stanford University 1977.

SINNHUBER R.

'Explaining and Justifying the Reasoning of Expert Systems:
A Review of Recent Work'

University of Sussex A.I. in Medicine Group Report AIMG-4,
1984.

STICKLEN J., CHANDRESEKARAN B. and JOSEPHSON J.R.

'Control Issues in Classificatory Diagnosis'

Proc. Int. Joint Conference on Artificial Intelligence-9,
1985.

SOWA J.F.

'Conceptual Structures'

Addison Wesley, 1984.

SWARTOUT W.R.

'A Digitalis Therapy Advisor with Explanations'
Proc. Int. Joint Conference on Artificial Intelligence-5,
1977.

SWARTOUT W.R.

'Explaining and Justifying Expert Consulting Programs'
Proc. Int. Joint Conference on Artificial Intelligence-7,
1981.

SZOLOVITS P. and PAUKER S.G.

'Categorical and Probabilistic Reasoning in Medical
Diagnosis' Artificial Intelligence Vol.11(1978),
pp 115-144.

TORSUN I.S.

'Knowledge Representation: An Overview'
Proc. Architecture of Large Knowledge-Based Systems
Conference, Manchester, 1984.

WEISS S.M., KULIKOWSKI C.A., AMAREL S, and SAFIR A.

'A Model-Based Method for Computer-Aided Medical
Decision-Making'
Artificial Intelligence Vol.11(1978), pp 145-172.

WELD D.S.

'Combining Discrete and Continuous Process Models'
Proc. Int. Joint Conference on Artificial Intelligence-9,
1985.

VAN RELEGHEM E.

'Separating Control Knowledge from Domain Knowledge'
Proc. ECCAI 1984, Pisa.

APPENDIX II

EXAMPLE QUESTIONS SUPPLIED BY UNILEVER

The examples in this Appendix illustrate how ADEPTUS can be used to produce useful responses to questions posed by industrial users of a coating selection system. Each question is given first in the natural language form as it was originally expressed. The internal ADEPTUS representation is then shown, followed by ADEPTUS' actual response. This is then rewritten expressed in natural language if necessary.

Example 1

What do you mean by a 'weak alkali' ?

-> (DEFINITION 'WEAK-ALKALI)

Answer:

***** Weak-Alkalis *****

are a class of **Alkalis**

with state default **liquid**

with pH range (7 10)

Example 2

What is the relevance of humidity in this context ?

(This concerns the system's request for an estimate of the humidity during the application of the coating.)

```
-> (GOAL-ORIENTATION '(ACTOR ADEPTUS) '(ACT place-value)
      '(OBJECT (environment-description humidity)) )
```

Answer :

```
((ACTOR ADEPTUS) has-goal
 ((ACTOR ADEPTUS) (ACT PREDICT)
 (OBJECT
 ((STICK ?C ?S) affected-by
 (environment-description humidity) ))))
```

i.e. ADEPTUS is trying to predict the way in which the coating and substrate stick together, and this is affected by the environmental humidity.

Example 3

What is a chlorinated rubber coating ?

-> (DEFINITION 'Chlorinated-rubber)

Answer:

***** Chlorinated-rubbers *****
are a class of **coatings**
with *adhesion* default (average)
with *colour* **all**
with *constituents* (**chlorinated-rubber-solvent pigment**)
with *state* one-of (**solid liquid**)
with *single-film-thickness* range (0 300)
with *defects* one-of (**pitting blistering cracking flaking**)
with *hardness* range ((extremely soft) (very hard))
with *flexibility* range
((extremely rigid)(extremely flexible))
with *finish* default **gloss**

Example 4

What is the environment after the drying process ?

(The question refers to a previously instantiated script, containing details of a particular coating and its behaviour. ADEPTUS is required to pick up the appropriate binding for the environment in the state following the STATECHANGE act in the **\$d-script.**)

```
-> (DEFINITION 'Corroding-Environment2)
```

Answer:

```
*** Corroding-Environment2 ***  
is an instance of Corroding-Environments  
with immersion.in default none  
with humidity (very humid)  
with macrolocation industrial  
with mechanical-damage abrasion2  
with microlocation outdoors  
with splashed-by default none  
with temperature default 20  
with vapour isocyanate
```

Example 5

Is bitumen suitable for a substrate with a poor surface key, used in an exterior industrial environment ?

(This question means that a **\$d-script** must be run for bitumen, in the environment described. The binding for the worn dry film is then accessed and forms the question concept.)

-> (JUDGMENT 'dry-film-bitumen6)

***** dry-film-bitumen6 *****

has very good *adhesion*

and has extremely good *single-film-thickness*

and has solid *state*

but has pinholes *defects*

and has chalked *finish*