The Knowledge Engineering Review, No. 6:1, 1991, 1–19

Explaining reasoning: an overview of explanation in knowledge-based systems

RICHARD W. SOUTHWICK

Department of Computing, Imperial College, London, UK

Abstract

There seems to be general agreement amongst those involved in KBS research that in order to be useful, a system must be able to explain its reasoning to a user. This paper reviews the development of explanation facilities in knowledge-based systems. It differentiates between explanation as a problem-solving process, and that which explains a reasoning process. This review concentrates on the latter, identifying and giving examples of three categories of reasoning explanation.

We then look at user requirements for explanation. What makes an explanation useful depends on the expectations of a user, which in turn depends on such issues as user background and system context. Several techniques are examined that have been applied to the problem of producing explanations that are appropriately structured and conveyed.

Finally, we discuss some of the work that has been done in describing theories of human discourse and explanation, and some issues that will become increasingly important for future explanation systems.

An extensive annotated bibliography is provided.

1 Introduction

In the fifteen years that researchers have been exploring how to encapsulate human knowledge in a computational model, several general principles have emerged. It is widely agreed, for example, that it is important in the design of knowledge-based systems to separate the knowledge base from the inference engine, and that knowledge should be represented in a uniform manner. One of the most important of these tenets is that a system must have the means to explain its conclusions to the user. This paper is concerned with the issues involved in building explanation facilities into knowledge-based systems.

The need for explanations is a fundamental one in the design of knowledge-based systems. First, a system needs to explain what it has done, to assure the user that the reasoning is logical and the conclusions sound. In a medical diagnosis system, no user would accept the conclusions of a computer system unless it could describe how they were derived. Second, good explanations may also persuade a user that a system's conclusion is appropriate. If a system produces an unexpected solution, a good explanation of that conclusion may convince the user of its relevance. Third, an explanatory facility could also be used as a training aid, by describing the system's domain knowledge and inference techniques. Finally, explanations are necessary as a debugging aid. The system designer needs to be able to trace the sometimes tortuous deductive path the system followed, and to be able to find and correct invalid inferences.

This paper begins with a discussion of the distinction between specific explanation and reasoning, or procedural explanation. We identify three categories of explanations used in explaining reasoning, review the historical origins of these ideas, and discuss their relative strengths and weaknesses. We then examine several explanation systems, and see how they relate to these basic explanation models. We then look at a user-centred view of explanation, and discuss a variety of techniques useful in designing good explanations. Finally, we examine some of the broader areas of research that will be central to the next generation of explanation systems.

2 Explanation of reasoning versus scientific explanation

Explanation has long been a topic of interest to philosophers, especially those interested in science and reasoning. Philosophical theories of explanation split the topic into two areas. One view treats explanation as a mode of reasoning. This kind of explanation is often referred to as *scientific* or *deductive* explanation (Hempel, 1965). Scientific explanations are concerned with the construction of theories to explain observable phenomena. If my dog comes in soaking wet, for example, an explanation of this fact might be that it is raining outside.

Explanatory hypotheses of this type require an *abductive* reasoning process (Draper, 1987b). In order to arrive at a theory, a person must first find some pre-existing model or schema, and try to interpret all data in terms of that model. This abductive leap from a small amount of data to a working hypothesis is risky, and studies have been made that show how people develop (often mistaken) theories about computer operation (Lewis and Mack, 1982).

A great deal of work has been done on abduction as a reasoning paradigm for generating explanations; one example is Josephson *et al* (1987), wherein a mechanism is presented that assembles hypothesis parts into a unified explanatory hypothesis in order to perform abductive inference.

This can be contrasted with the view that to explain a process or state, one must give an account of the way in which that process or state is achieved. For a system that reasons, explanations provide insight into how that reasoning is accomplished—an explanation is *about* reasoning. We shall call explanation of this type *procedural* or *reasoning* explanation.

Thus, we can make a distinction between these two views of explanation: although the generation of a hypothesis to explain an observation may be a useful reasoning technique, a procedural explanation may be necessary in order to understand *how* this hypothesis was arrived at.

The field of scientific explanations and hypothesis construction is large and growing. In this paper, however, we restrict our attention to procedural explanations, looking at systems that can explain their reasoning, regardless of the reasoning methodology employed.

3 Categories of explanation

In the historical development of reasoning systems capable of explaining their reasoning, three main approaches to explanation have emerged. These can be summarized as follows.

1 *Reasoning trace explanations:* these are explanation at the system level, able to give information about the contents and structure of the knowledge base. They explain why a conclusion was reached, or a decision made, by describing the reasoning steps that led to the conclusion. Typical of this category are the standard HOW and WHY rule trace explanations.
2 *Strategic explanations:* rather than explaining a result by listing rules used, these explanations describe the strategy employed by the problem solver. Strategic explanation give the user an insight into the problem-solving methodology.
3 *Deep explanations:* deep, or model-based explanations *justify* system results by linking them to a deep, causal model. Thus, deep explanations attempt to give the underlying reasons for an action or state.

While reasoning trace explanations rely only on the formulation of knowledge that comprises the knowledge base, the other two explanation types require additional knowledge. A system that can give strategic explanations must be able to reason about its own activity, which may require knowledge about the ordering of problem solving tasks, for example. Deep explanations obviously require a great deal of information in the form of a causal model of the domain.

To see the role that these explanation types play, consider a diagnostic system for car maintenance. Suppose that the system is explaining its conclusion that a clogged fuel filter caused the engine to die. Such a system might give the following explanations, corresponding to the three categories

1 *Reasoning trace:* "You told me that the engine spluttered, and I know that if the engine coughs, then the filter may be at fault."
2 *Strategic:* "There are three engine subsystems to check. I checked the fuel system first, because the symptoms indicated the likelihood of a fuel system problem."
3 *Deep model:* "A clogged fuel filter prevents petrol from reaching the carburettor, thus causing engine failure."

In a rational reconstruction of MYCIN (Shortliffe, 1976), Clancey lays the epistomological groundwork for these explanation types (Clancey, 1983). He characterizes knowledge as filling one of three roles: that of structure, strategy or support. Structural knowledge is represented by the underlying stuctural framework, with which relations in the domain knowledge may be categorized. This knowledge is available for reasoning trace explanations. Strategic information is used in formulating a plan for ordering goals, and can be used for strategic explanations. Support knowledge consists of the justification for a rule, and is necessary for deep explanations. Clancey provides an illustration of these concepts using the following rule from MYCIN:

If the patient's age is less than seven years, then don't prescribe tetracycline.

The structural component of this rule categorizes the rule with respect to the consequences and conditions of the rule. The strategic component, which indicates when the rule should be applied, can be summed up as "Consider side-effects before prescribing therapy". And the support knowledge, in this case, makes clear what would otherwise be a somewhat baffling rule. The justification for this rule is that tetracycline will discolour the teeth of young children. Although Clancey couched his exposition in terms of a rule-based system, the same analysis applies to any representation scheme. Regardless of how the knowledge is packaged, these knowledge roles are still pertinent.

Because an explanation of a reasoning process requires some reasoning *about* reasoning, it may be termed a *metalevel explanation.* The role played by these explanations can be demonstrated through a simple example. Suppose that we are representing knowledge in the form of logical relations, and that our inference method is deduction. Let us take the following sentences as the knowledge base.

$$P \qquad \begin{matrix} P \rightarrow Q & \quad Q \rightarrow S \\ P \rightarrow R & \quad R \rightarrow S \end{matrix}$$

With this knowledge base, S may be derived as a conclusion. If we are interested in scientific explanation, we may take S as an observation, and produce P as a hypothesis that "explains" S.

An explanation of the reasoning, however, might begin with a proof trace: "S because P, $P \rightarrow R$ and $R \rightarrow S$". A strategic explanation would focus on which rule was used and why: "$P \rightarrow R$ was chosen due to a left-right selection rule". A deep explanation requires additional information, perhaps from another database: "$P \rightarrow R$ because $P \rightarrow P'$ and $P' \rightarrow R$". Finally, the inference method being used is fundamental to understanding an explanation: "P and $P \rightarrow Q$ therefore Q, by *modus ponens*".

4 Reasoning trace explanations

The origin of reasoning trace explanations was MYCIN (Shortliffe, 1976), the prototypical rule-based backward chaining system. The original system had, as a debugging aid, the ability to print out an English translation of the rule it was currently considering. It became apparent that this was insufficient, and that an explanation facility was needed. The adopted solution was to keep a trace of the rules used during the deductive process, and to use this information to explain the system's actions (Davis, 1982). The user could issue WHY and HOW queries to examine the execution trace and the rule base. When asked questions by the system, a user could in turn ask WHY the question was being asked. A WHY question ("Why are you asking me this?") traced the reasoning chain

upwards, and presented a trace of the rules which led to the system's request. A HOW question ("How did you get the answer?") explained the system's conclusions by descending a successful trace of the reasoning network. This was a simple and powerful method for giving the user a window into the knowledge base, and is still widely used, especially in expert system shells (Hammond and Sergot, 1983).

Many commercial shells, for example, provide explanation facilities based on minor variations of the reasoning trace paradigm. The limited nature of these facilities is due in part to the required generality of such systems. KEE, from Intellicorp, provides basic HOW/WHY explanations, augmented by a sophisticated graphical interface for displaying knowledge base relationships and inference traces. Because the capture of the additional knowledge necessary for more advanced explanations is not designed into these shells, explanations that rely on this information cannot easily be generated.

When using an expert system, a user has three opportunities to interact with the system: during the consultation, to ask for a reason for the system's behaviour; after the consultation, to request a justification for the results; and independent of a consultation, when the user may ask general questions about the domain. This observation formed the basis for an extension to MYCIN's explanation capabilities (Scott *et al.*, 1977). The improved explanation capability was able to handle questions about all relevant aspects of the system's knowledge and actions. It was split into two modules: a reasoning status checker (RSC), used during the consultation, and a general question answerer (GQA), to answer questions about the current state of the system's knowledge. The RSC incorporates HOW and WHY commands to examine the execution trace. The GQA, however, uses natural language routines to allow the user to ask about the system's conclusions, the static knowledge base, and any deduced facts. The GQA comprises a set of question-answering specialists, each of which knows how to answer questions of a specific type. A user's questions are reduced to a set of terminal words, then classified according to type.

Scott gives us some examples of the types of questions about consultations

What is ⟨val⟩ of ⟨cntxt⟩: "To what class does Org-1 belong?"
How do you know ⟨val⟩ of ⟨cntxt⟩: "Why didn't you think the site was urine?"
How did you use ⟨val⟩ of ⟨cntxt⟩: "Did you consider that X is a compromised host?"
Why didn't you find out about ⟨val⟩ of ⟨cntxt⟩: "Did you find out about the CBC of Org-1?"
What did ⟨rule⟩ tell you about ⟨cntxt⟩: "Why didn't you use rule 189 for Org-1?"

The parsed question is then handed to a specialist function, which produces an answer by checking the execution trace or knowledge base, extracting the necessary information, and filling in an answer template.

In an effort to extend the scope of reasoning trace explanations in a general way, Wick and Slagle (1989) apply a journalistic analogy to explanations. Their system, Joe, can explain knowledge base relationship in terms of the "5 W's" of new reporting—Who, What, Where, When, Why (and How). Their goal was to extract the greatest possible amount of information from a reasoning trace, and allow the user immediate access to that information. In a similar way, this type of information is presented in the 'lead', or first paragraph of a well-written news story. Supplementary forms of explanation may follow (as in the body of a news story, to pursue the analogy), if the necessary information is available. What, How and Why questions present standard reasoning trace information. To answer Who queries, the system records the author of a data entry. When questions are answered by time-stamping each data value. The use of time stamps allows the system to determine relational links temporally, rather than requiring an explicit representation.

4.1 Problems with the MYCIN paradigm

While extended reasoning trace explanations are far more flexible and usable than the original MYCIN explanation facility, they still have major shortcomings. Simple system-level explanations

do not take user's needs or abilities into account, as the same answer is given to all users. Different users have different interests and points of view; a novice user will not be satisfied with the same explanation as that given to the system designer.

Since the explanation is based on an execution trace of the rule base, the quality of the explanation depends entirely on the formation of the rules used. Variable names used in the code appear in the explanation, for example, so must be meaningful. Even given an understanding of the domain, some knowledge of the structure of the rule base seems to be necessary for the comprehension of rule-trace explanations. In addition, most expert systems have added many extra rules which are not part of the domain knowledge. These rules may perform system book-keeping, or act as screening clauses to improve interactive behaviour. Rules included simply to improve efficiency will still appear in the explanation, although they are not necessary for understanding of the reasoning process.

The major failing, however, was described by Clancey when he attempted to use the MYCIN rule base in a tutoring system (Clancey, 1983). He began with the reasonable supposition that a knowledge based used for deduction could also be used for teaching, but soon discovered that this was not the case. The problem is that an execution trace does not provide sufficient information, because there is no justification for why a conclusion logically follows from a premise. A set of rules represents the knowledge of an expert in a compiled form, with all the deep associational links removed. In this sense it is flat, and so cannot provide a justification for its actions. To sum up, shallow rules give shallow explanations.

What then, do we expect in an explanation? This is the question that post-MYCIN systems must try to answer. It is clear that a system must produce reasons for a conclusion, action or state of affairs, in such a way that it is justified. Simply describing the behaviour of a consultation process (using an execution trace) is not sufficient. Moreover, a system must be able to tailor its explanations to meet user's expectations. Satisfying this requirement could mean anything from simply removing extraneous detail from an explanation to building a comprehensive model of the beliefs of a user. Finally, an explanation is useless unless it empowers the user in some way. Some explanations may cause a user to feel happy with a system's conclusion, without conveying the necessary information required to be able to *refute* that conclusion. The user of a system must understand the conclusions well enough to be able to take responsibility for acting on them. Thus, the ability to act on or refute an expert system's conclusion is a measure of the efficacy of the explanation.

5 Strategic explanations

Problem solving requires a certain amount of strategic planning. The designer of a reasoning system uses this problem solving strategy when constructing a knowledge base. The order in which rules are written, for example, affects the behaviour of a rule-based system, and very often system behaviour is controlled through rule or task ordering. The metalevel control information, however, is not made available to the object-level system, and although implicit in the object-level structuring, is not available to the user. To provide strategic explanations, this control information must be explicitly represented.

5.1 NEOMYCIN

In attempting to reconfigure MYCIN for teaching (Clancey, 1981), Clancey encountered many limitations in MYCIN's knowledge base and explanation abilities. In NEOMYCIN, one of the design goals of the system was to be able to explain the strategy employed in solving a problem (Hasling *et al.*, 1984). This requires an explicit representation of the problem-solving process, which is encoded in a body of metaknowledge that describes the strategic decisions the system would make. This knowledge is structured into tasks, which consist of metalevel goals and

subgoals, and metarules, which are the methods used for performing tasks. Using this additional knowledge, NEOMYCIN is able to explain its current strategy during a consultation. The user can request a strategic explanation whenever the system queries the user for information. If the user asks "Why is it important to determine whether Mary has been hospitalized recently", the system can respond with a more abstract explanation: "We are trying to round out the diagnostic information by looking into the past medical history".

5.2 Topic explanations

In another example of a strategic explanation approach (Southwick, 1987), it is argued that greater understanding of the solution presented by an expert system requires an understanding of the process that resulted in that conclusion. A system is described that makes explicit the logical structure of the rule hierarchy by notifying the user of the topic under consideration. The aim is to explain the process dynamically while the consultation is proceeding, so that at the end of the consultation, the user has received both an opinion, and a description of the problem-solving process.

This is achieved by segmenting the knowledge base into topics to be considered, and presenting an ongoing trace of the topic goals that have been satisfied, and that are currently being explored. Without this type of additional information, the user is often overwhelmed by a stream of seemingly unrelated questions, asked by the system in response to missing information. When this information is supplied, the result is a dialogue with the system, where the user is notified of the current area of investigation, and the result of attempting to complete a topic.

We are now trying to show John unemployed.
    Is it true that John is currently-employed? NO.
    Is it true that John is available-for-employment? YES.
    :
John unemployed SUCCEEDS.
We are now trying to show John eligible-for-benefit. There are two ways to satisfy this goal:
    1. John entitled-to-pension.
    2. John entitled-to-supplementary-allowance.
    :

6 Deep explanations

When knowledge, elicited from an expert, is represented in rule form, information is lost. The rule consists of preconditions and a conclusion; the reason for the conclusion following from the precondition has been compiled out. Although the justification for the inclusion of a rule was used in the design of the system, this knowledge is unlikely to appear anywhere in the rule base.

This missing information is often referred to as a "deep model" of the domain, which explicitly represents relations that are only implicitly represented in a shallow or compiled knowledge base (Klein and Finin, 1987). There has been much interest in the development and use of deep knowledge in reasoning systems (Chandrasekaran and Mittal, 1983), leading to a surge of interest in recent work on qualitative reasoning (such as de Kleer and Brown, 1985).

Research into deep modelling is of interest to designers of explanation systems because of the inadequacy of explanations based solely on shallow knowledge. An explanation system that has access to a deep model can provide explanations that are intuitively more satisfying, since they relate to the deeper concepts that underlie the domain model. A financial modelling system, for example, might give deep explanations based on economic theory. Once a deep model representation of a domain has been captured, the techniques used for presenting explanations are similar to those for reasoning trace and strategic explanations.

6.1 XPLAIN

The XPLAIN system (Swartout, 1981) was designed to capture the deep support knowledge necessary to explain and justify a system's behaviour. Swartout tackles two problems with his design: explanation production and automatic programming. His approach is a simple but powerful one. Rather than try to encode whatever deep knowledge seems necessary, he has built a program generator that can write the domain program itself, whilst recording the links between the principles used in the domain, and the body of factual knowledge. To build a system, then, one must first specify the domain model, which comprises descriptive facts and causal relationships important in the domain. This model is entirely declarative, so that the system can use the same piece of knowledge in different ways. Procedural knowledge is represented as a set of domain principles, which are methods and heuristics that specify how something should be done. Each domain principle has three parts: the goal (what the principle can accomplish), the domain rationale (the links to the domain model that specify the applicability of the principle), and a prototype method (what to do). The application program is constructed by the Writer module, which starts with a top-level goal and, using the domain model and principles, gradually refines it by creating more specific subgoals until the level of system primitives is reached. A trace of this refinement process is retained for use by the object-level program. When the program is executed, explanations are produced using this refinement record, along with the domain model, domain principles, and an execution trace.

This approach offers many advantages. The strict separation of the domain model and procedural knowledge makes explicit the strategic, structural and support knowledge in the domain. The maintenance of such a system is greatly simplified, since its production is carried out automatically (Neches, 1985).

7 User-centred explanation

We have been looking at various types of explanations, and the knowledge required to implement them. Let us now shift our attention from knowledge representation to look at explanation from a user's point of view. Work in this area concentrates on varying the form and content of explanations according to the perspective of a user, and on extending the flexibility of explanation systems. There are several issues that must be addressed, including:

- What kind of questions can a user ask?
- In what form are explanations presented?
- How are user's individual requirements catered for?

7.1 Customizing explanations

A good deal of attention has been given to the problem of making systems more responsive to individual requirements. First attempts to improve the quality of reasoning trace explanations focused on regulation of levels of detail—the granularity of the explanation. A user can easily become overwhelmed by the amount of information that a reasoning trace can generate, especially when rules are large, or deeply nested. This large amount of output is due to the fact that reasoning trace explanations are necessarily based on the structure of the knowledge base.

There are two issues to be considered here. First, users with different levels of experience want to see different details of the knowledge base. Second, expert systems contain "computer artifacts"—rules which are included to increase the efficiency of the system, but which are not a necessary part of the reasoning process. System designers have tried to come up with ways to limit the amount of detail, to filter out computer artifacts, and to focus explanations towards a particular point of view.

In TEIRESIAS (Davis, 1982) the certainty factor associated with a rule is used as a crude

measure of its information content. Rules with high certainty factors (e.g., definitional rules) are taken to be common knowledge, and so are regarded as having a low information content. The user can then specify the amount of detail used in an explanation by setting a threshold value. If the negative of the log of the certainty factor is above this threshold, the rule is not included in the explanation.

In BLAH (Weiner, 1980) the measure of detail corresponds to the depth of the reasoning tree. At a certain level of detail, the user will be told of all rules used to a specified depth, eliminating details of branches further down. This scheme requires that the reasoning process be represented in a hierarchical tree format, with each goal or procedure being broken into subgoals at a lower level of detail. In addition, the name of a procedure call must be a good description of its action, since at the lowest point of the specified detail level, the name is all that is seen by the user.

Wallis and Shortliffe (1982) represent the domain as a causal network. Concepts in the causal model are rated according to complexity and importance. The user can then select the level of difficulty desired, and the system will leave out the concepts in the reasoning chain that are not understood. UMFE (Sleeman, 1984) builds on this notion by interactively setting the difficulty level. It begins by asking the user whether a particular concept is understood. If it is, it is assumed that all sibling concepts with the same importance rating are also understood.

In XPLAIN (Swartout, 1983) the computer artifact problem is handled by attaching a viewpoint to steps in a domain principle. The viewpoint indicates to what type of user the step should be explained. When the explanation is being generated, unnecessary or uninteresting steps are filtered out according to the user's viewpoint.

7.2 User models

It has become increasingly clear that in order for a system to be able to produce natural explanations, it must have information about the concepts understood by a user, so that explanations can be phrased in those terms. First attempts at customizing explanations were *ad hoc* solutions, but since then research into methods of representing a model of a user's knowledge and belief has influenced the design of explanation systems. User models can be classified along three dimensions (Rich, 1983). They are (1) single canonical user versus a collection of models of individuals; (2) explicit models versus inferred models; and (3) models of long-term versus short-term characteristics. This framework can be used to describe various approaches to user modelling.

In early attempts at computer tutoring systems, the user model typically consists of the responses made by the student. In WEST, for example, the system infers the user model by monitoring the progress of the student (Burton and Brown, 1979). Wrong responses indicate poorly understood concepts, and cause the system to focus on those concepts.

There can be a problem with inferred models, however. If the system updates the model on the basis of the user's performance, and then shifts its operation according to that model, the user's context might suddenly change, without any clear indication of why this is so. If a user desires a particular style of interaction, some experience with the system will teach him what kind of responses to make in order to get the desired behaviour. The result of this, however, is that the user is no longer being modelled, but is trying to second-guess the system, thereby undermining the point of a user model.

One approach to creating individual models is to create a set of stereotypes, in terms of interesting attributes (Rich, 1979). User profiles are based on a stereotype, but individualized according to the particular details of the user. A stereotype can be used to generate many facts about the user without having to specifically gather them.

One of the problems faced in the design of a good user model is that of keeping track of the current beliefs of the user. A truth maintenance system (TMS) (de Kleer, 1986; Southwick, 1989) could be employed to perform this task. This suggestion was first implemented in BLAH (Weiner, 1980) which uses a TMS to determine the set of currently believed assertions. Attached to each assertion is a justification for it, consisting of a list of assertions justifying belief in the assertion.

When BLAH generates an explanation, it uses this knowledge about what the user already knows to delete uncessary information.

7.3 Question types

One method of limiting the scope of possible user input while still giving satisfying explanations is to specify the type of questions that might be asked by a user, and designing the system to be able to handle those types of questions. We saw this approach used in MYCIN's general question answerer (Scott *et al.*, 1977), and a similar approach is used in many systems (e.g. XPLAIN). A rigorous investigation was made by Lehnert (1978), who described a hierarchy for question classification. In designing a system that would understand simple news stories, Lehnert recognized that understanding is best demonstrated by the ability to answer questions about the domain. In this system, natural language questions are divided into a question type, and a question concept. Lehnert identified 13 question categories that an expert might be expected to answer. When Lehnert's question categories were modified and implemented for expert systems (Hughes, 1986), it was discovered that the traditional HOW and WHY questions covered at most three categories, highlighting their restrictive nature. WHY questions correspond to Lehnert's 'Goal orientation' category, since we are interested in determining the current goals of the system. HOW questions fall into the 'Enablement' or 'Causal antecedent' categories. We are left with no satisfactory answers to procedural questions (How do you X?), comparative questions (Which is better, X or Y), expectation (Why didn't X happen?) or any of the other categories in Lehnert's classification scheme.

7.4 Explanation grammars

In order to create useful explanations, knowledge is needed about both the reasoning process being explained, and the intended recipient of the explanation. In addition, a system also needs information about the form and structure of an explanation in order to construct the best explanation.

One approach to defining the possible structure of explanations is a grammar-based approach. Cawsey (1989) proposes a grammar for the production of explanations of electric circuit behaviour.

Explain → Describe, Explain Behaviour, Summarize.
Explain → Type of circuit, Summarize.
Describe → Type of circuit, Function, Describe Components.
Describe Components → Function, I/O behaviour.
Explain behaviour → Explain dependencies, Explain I/O.
Summarize → Function, I/O behaviour.

According to these rules, an explanation can consist of a description of a component, followed by an explanation of the component behaviour, followed by a summary. One form of a description is a description of the type of circuit, its function, and a description of its subcomponents.

The full grammar, of which this is a fragment, states the form of a legal explanation. This grammar is non-deterministic—there are many legal forms, allowing explanation production to reflect user requirements and system context. With this approach, the problem becomes one of controlling the explanation generation procedure. Typically, a planning system is needed that ensures that necessary preconditions are met before committing to a particular branch of the grammar.

7.5 Text generation

As explanation systems have become more sophisticated, a need has arisen for output in clear, easy to read English. Earlier approaches have used natural language templates where the key words

were filled in by extracting system parameters, but this lacks the flexibility of true text generation. The problem of text generation has until recently been ignored by natural language researchers, who have occupied themselves with attempting to understand input, rather than producing output. But research (McDonald, 1982; McKeown, 1985b) has resulted in generators with extensive grammars and explicit representations of English text. These systems are able to automatically handle the intricacies of English construction and make sophisticated linguistic decisions. As expert systems increase in size and complexity, *ad hoc* generation methods will soon fail to cope. Text that is the result of recursive problem solving methods, for example, can become difficult to handle properly in large systems. An interface between an expert system and a text generator would allow the system to describe the desired text in some high-level fashion, and leave the details of the generation plan to the text system.

7.6 Graphics

Explanations need not be restricted to text. Many applications lend themselves to the graphical representation of the state of the system. If the structure of the domain can be described graphically, (through use of a proof tree, network diagram, etc.), the system's metalevel behaviour and control can be conveniently depicted. Alternatively, if the domain permits it, object level graphics can provide a deeper understanding of the concepts and procedures represented in the domain.

Recent developments in hardware and graphic capability have permitted major improvements in the way reasoning can be illustrated. The LOOPS programming environment, for example, enables the display of a program's structure, and allows the user to browse through a complex structure, requesting information on selected items (Bobrow and Stefik, 1983). This capability is exploited in GUIDON-WATCH, an instructional program that sits on top of NEOMYCIN (Richer and Clancey, 1985). This system is a complex, interactive system for browsing through the knowledge base and watching the reasoning process as it progresses. Both the dynamic task tree (the structure of the domain knowledge) and the task stack (list of current goals) are represented, giving the user a view of the current task, and a window into the structural context of the task.

Some domains are inherently pictorial, and are best presented graphically. The explanation of the location of a fault in an electrical system, for example, can be clearly made with the display of a circuit diagram where different components are lit up as they are considered by the system. This is the approach used in the Steamer project (Hollan *et al.*, 1984). Steamer is an intelligent training system, designed to assist in propulsion engineering instruction. The key idea of Steamer is that of linking a colour graphics interface to a simulation model, to create an inspectable simulation which reflects both the state of components in the system, and allows the manipulation of system elements. Steamer produces dynamic graphical explanations using a set of icons that describe behaviour that would otherwise be opaque and difficult to describe in text. Icons that change dynamically represent the state of the system and provide a graphical description which is similar to the qualitative model of an expert.

8 The next generation

As expert systems and their explanation facilities become better understood and more sophisticated, researchers have begun to take a broader view of the issue of explanation. People want a more natural, flexible interaction with knowledge based systems, and chafe against the restrictive style of existing systems. One result of this is a shift in the definition of explanation from simply describing the behaviour of a program, to the type of explanation that attempts to convey the understanding of a set of concepts. This is the direction that has motivated each successive improvement in the design of explanation capabilities.

8.1 Role of human explanation

When studying the broader issues involved in explanation, the natural place to start is the analysis of human interaction. Much attention has been paid to the plan formation and goal satisfaction that occurs during human discourse. Although this has long been the territory of linguistics, it has been adopted by those studying natural language understanding, and, more recently, explanation generation.

In Allen and Perrault (1980), the problem addressed is that of inferring the goal of a speaker, given a discourse segment. Allen discusses several types of speech acts, and develops a representation of speech and domain plans. He uses plausible inference rules to derive information about the current state of a speaker's goals and beliefs. These techniques are employed in the Advisor system (McKeown *et al.*, 1985a), which derives the user's goal from a sequence of utterances. The derived goal is then available for use in generating an explanation, in natural language, that matches the point of view of the user.

In addition to being well directed, explanations should also be complete. Human communicants take care that their responses will not mislead the listener. A respondent is expected to provide whatever extra information that he recognizes to be necessary, even though that information was not explicitly requested. Not providing the information is deemed uncooperative. In other cases, silence on the part of the respondent can mislead the listener. If a listener expects a speaker to convey a piece of information if it is known, and the speaker fails to do so, the listener assumes that it is false (the "closed world assumption" of discourse). In cooperative man–machine interaction, if the system believes that a planned response might mislead the user, it must block that conclusion by modifying its response. In Joshi and Webber (1984) a formal analysis is undertaken of the type of information that should be included in a response to a question, so as not to mislead the questioner. The question they take as their example is "Can I drop CS557?", representing the goal of dropping a university computer course. A set of cases are identified where additional information should be presented. For example, if the enabling conditions of the goal fail, but there is another way of achieving the goal, that should be included ("No, but you can audit it instead"). Or if there exists a better method of achieving the underlying goal, it should be mentioned, even though it is not explicitly stated in the question ("Yes, but perhaps you should take an incomplete instead").

In a study of the structure of human plans, as represented in natural discourse, Linde and Goguen (1978) analyse transcripts of a planning session, and show the regular structure of such discourse as a tree-structure plan. Extending this approach to the analysis of discourse, a model of the structure of human explanation is presented by Goguen *et al.* (1983) as a precise and computationally effective foundation for artificial explanation systems. Natural explanations are represented in a tree structure whose nodes correspond to the three major types of justification: giving a reason, giving an example, and eliminating alternatives. The production of an explanation, during a discourse segment, is represented by a sequence of transformations on a tree as new material, with either semantic or structural content, is introduced by the explainer. Focus of attention is represented by a pointer into the tree, and a shift of focus by pointer movement. This pointer marks the part of the explanation that is currently being developed, and gives an indication of where the next development will occur.

Study of the structure of human discourse has stimulated an interest in ethnomethodology and sociolinguistics (Gumpertz, 1972). Ethnomethodology attempts to capture and represent the shared patterns of communication and common understanding of members of a society. In any community, there is a large body of unstated assumptions which are understood by all, which colour the structure of communication, as much cultural information is implicitly conveyed.

8.2 Interactive explanation

An assumption that underlies much of the work done on explanation is that there is a single, perfect explanation for any given situation and user. A great deal of the interest in user modelling, for

example, is prompted by the need to set parameters for explanation in advance, so that the "best" explanation will always be given. Work done on categories of question types also makes the assumption that we can pre-calculate the form and content of explanations. But it may be that an interactive process is the only way to arrive at a truly useful explanation.

Draper (1987a), for instance, argues that there is little linguistic marking of the role of an utterance, so that it is difficult to determine what the user's goal is in asking a question. For example, the question "Where does the train leave from?" will require a different answer for a commuter, than it would for a tourist. Since the person asking the question often doesn't know the exact type of answer needed, the explainer must determine the nature of the request from the context of the discourse, and from estimates of the enquirer's knowledge. Consequently, Draper concludes that taxonomies of question types have little promise, because they fail to measure the differences that exist between two sets of beliefs, and to accept that the goal of explanation is to reduce that difference.

One of the observations that has emerged from the study of natural explanation is that the process of human discourse is essentially a cooperative one. Asker and explainer must work together to determine what exactly the asker's goal is. Typically, this puts the burden on the explainer, who must try to deduce the asker's goal in the context of societal conventions and evidence of the state of the asker's knowledge and belief. The asker has a responsibility to cooperate with the explainer, however, in negotiating the terms of the consultation.

This seems to suggest that the standard expert system paradigm is inadequate. Expert systems typically collect data and produce a solution. They can only work on a very restricted problem set, and demand a level of expertise from the user, who must be able to pose the initial query properly. Good performance in this does not guarantee user acceptance, however.

An alternative to the domineering and unresponsive style of expert systems is that of critiquing. In a critiquing system, the user presents his idea of a solution to a problem, and the system critiques it using its expert knowledge base. In the ATTENDING system (Miller, 1984), the domain is anaesthetic management. Given a user's plan, the system can produce possible alternatives, weeding out the high-risk options, and produce a point-by-point comparison of the two plans. The polished text of the critical discussion is built up out of template parts as the representation network is traversed during the critiquing process. The critique has several parts. First, there is an initial discussion of the underlying principles involved in the case. The body of the critique mentions any positive aspects of the plan, and then possible problems in the plan are singled out, and alternatives are presented.

A descendant of MYCIN was also adapted to critique user plans (Langlotz and Shortliffe, 1983). The ONCOCIN (cancer therapy) system accepts, analyses and critiques a physician's plan. An interviewer module collects data about the case and the proposed plan, whilst a reasoner module uses a rule-based reasoning system to arrive at a recommendation. The system employs hierarchical plan analysis to determine where the two plans differ. Because the domain representation is hierarchical, it is possible to find the most general set of differences. Significant differences are explained to the user by building a list of parameters that differ at the most general level. The user may request more detailed explanations of any items on this agenda list, and if in that explanation, further parameters are encountered, they are added to the agenda to be explained.

Alty and Coombs (1984) also take the position that human experts are most often called upon to assist other experts in extending and refining their understanding of a problem. Traditional systems produce solutions to well-defined problems, without taking into account the experience of the user. Human experts, on the other hand, are often called upon to provide assistance to other experts, at the junction of their respective fields. The expert helps a colleague to decide what questions to be asked and how to look for answers.

In a system designed to implement these ideas, the problem domain was that of an aid to novice Prolog programmers. The user enters a problem in the form of a Prolog program, and then describes the action of the program. The system checks the user's idea of how the program will

execute against its own, and tells the user of any discrepancies, so that the user can supply a new explanation.

8.3 Explanation by example

Techniques useful for reasoning by example (Rissland, 1983) can be exploited in explanation systems that present examples in response to the user's request for information. Giving examples is a common method of explaining things in human discourse, and provides an easy means to understand concepts or instructions. Rissland uses a taxonomic representation of example types to classify examples in the domain, and to determine their usefulness to the user. Knowledge of the user, his environment and his task provide constraints in the selection and construction of examples to be presented, resulting in customized explanations (Rissland *et al.*, 1984). In an on-line help system, for example, a record of the user's past actions can be used as a guide for selecting an appropriate example of the current task. If someone requests help in deleting a file from a print queue, the example is given using the contents of the actual system queue.

8.4 Tutoring

Tutoring systems occupy a position at the far end of the spectrum of explanation systems, since their entire concern is imparting understanding of the structure and problem-solving methodology of a domain. Other explanation systems must concern themselves with these issues, but in tutoring systems, they are paramount. We have seen how user models have played an important role in tutoring systems. User modelling is especially important in training systems because no assumptions can be made about the expertise of the user. This points to one of the problems that Clancey had with the MYCIN rule base (Clancey, 1981). In a consultation role, an expert system assumes that the user has an understanding of the structure of the domain, and explanations are often more to remind the user of rule than to impart understanding of the rule. But in practice, this is often not the case.

8.5 Separation of knowledge bases

In order to understand an explanation, people require that it be related to their own model of the domain. An explanation couched in concepts or methods foreign to a user will not satisfy. Humans can tailor their explanations by relating their knowledge to the experience of others. It is not even necessary for such explanations to be completely true, according to the knowledge of the expert, as long as they are useful in the domain model of the user. The notion of centrifugal force, for example, is taught to children to explain commonly observed phenomena. Since it is a useful explanation, and close to the conceptual model used by children, it is not necessary for the physicist to explain that centrifugal force is not a true force at all.

The problem is that the computational model used by expert system processes is significantly different from that used by humans (although it is more consistent). Not only is the domain knowledge different, but different reasoning strategies are used. Humans routinely reason by means of an inductive leap, forming hypotheses on the basis of incomplete information. An expert system, however, generally does an exhaustive search through its knowledge base. To respond to a user in his own terms, then, would require the representation of that inductive jump.

One solution to these problems is to have more than one representation of the knowledge used. An efficient and complete knowledge base is used for reasoning, and another, based on user concepts, for explanation. The BLAH system (Weiner, 1980) takes steps in this direction, by segmenting the knowledge base into system's and user's views. The system reasons using one set of information, and explanations generated by the other.

9 Conclusion

As the design of computer explanation systems has evolved, we have seen a steady movement toward more flexible, cooperative systems. We are moving from an expert system that produces a solution as a result, to a consultation system where the entire consultation is the result. Explanation has progressed from the verification of an expert system's solution, to become the solution itself. Essential to this progression is the ability to identify and represent the types of knowledge that need explaining, but which have not been necessary for efficient reasoning. We are still not sure how to represent the links between such deep knowledge, and the procedural or definitional knowledge that is typically extracted from experts and encoded in expert systems. In addition, we don't know how to relate different conceptual models in a domain. In natural explanation, the explainer chooses a conceptual model that he thinks will mesh with that of the listener, and using that model, decides how much detail to include. Explanations can thus be given on several different conceptual levels, each with varying amounts of detail. Whereas existing systems attempt to regulate the amount of detail according to user's viewpoint, what is really needed is a meta-model of the domain, which comprises a set of models employed by various users of the domain. Each model in this set may have its own knowledge base, representing principles in the domain according to its view. As an example, consider explaining by analogy. When someone explains electrical current in terms of a water system, using pipes, pressure and flow, he has chosen a conceptual model that is couched in terms that the listener will understand.

In designing computer systems, much work needs to be done on understanding how different conceptual models are related. If we do explain by analogy, how can we choose the appropriate analogy, and how do we know where the weak spots are, when switching to another model becomes necessary? When physicians talk about the nature of light, they employ either a particle theory or wave theory. How can we decide which model is appropriate? These are examples of the types of questions that must be answered.

10 Summary

The evolution of explanation systems has progressed from rule traces which were not much better than debugging aids, to systems that can generate text to explain concepts in terms that a user can understand. The evolutionary process has followed a pattern that seems to be repeated for any inventive process.

The first attempts at design are driven by the need for a solution to a particular problem, and generally consist of *ad hoc* methods. As people become more familiar with the problem domain, the shortcomings of existing solutions became clear. Particular problems with the original systems are targeted and attacked, in an attempt to improve performance. Finally, the system is reconstructed, using whatever lessons have been learned. This reconstruction is often the source of a more formal theoretical basis, which supplies a conceptual framework or taxonomic representation. Finally, new methods are suggested by assimilating techniques and methodologies from other fields.

This same process has occurred in explanation research. Many of the topics that have been discussed, and approaches that have been suggested have grown out of dissatisfaction with current methods. We saw the goal of customising explanation expand into efforts to model the state of a user. The production of the text of explanations has benefited from research done on natural language text generation. Tools and techniques from other areas of research have been applied, such as the use of question types, truth maintenance systems, etc. And finally, the entire endeavour is at the point where fundamental questions can be asked about the nature of explanation, both natural and artificial, in an attempt to begin to formalize the problem. We may discover in the future, in fact, that the ability to explain is so essential, and so wrapped up with issues of representation and reasoning, that explanation ceases to be valid as a separate research topic.

11 Acknowledgements

Thanks are due to my colleagues in the Logic Programming Group at Imperial College for their helpful discussions. Special thanks to Chris Evans and Bob Kowalski for their comments. This paper also benefited from comments made by reviewers on an earlier version.

References

These references have been organized by topic to enhance their tutorial value.

Explanatory systems

Clancey, WJ, 1983. "The epistemology of a rule-based expert system: a framework for explanation." *Artificial Intelligence* **20** 215–251. Knowledge can play one of three roles: structure (relations and structure of data); strategy (procedure for applying rules); and support (justification for applying rules; deep knowledge). Mycin's rules have compiled out much of this information and as a result, although optimized, the rule base is difficult to maintain, non-transferable, and not useful for explanation. Strategic, structural and support knowledge should be made explicit.

Davis, R, 1982. "Teiresias: applications of meta-level knowledge". In: R Davis and DB Lenat (eds.), *Knowledge-Based Systems in Artificial Intelligence* McGraw-Hill. In his chapter on explanation, Davis views explanations as being necessary to monitor the performance and output of a program. Information must be detailed, but not too detailed, and must be complete. The program keeps a trace of rules that it uses during the deductive process. HOW and WHY commands are used to move up and down the reasoning chain. The confidence values attached to rules are used as an information metric to assist in providing the right level of detail in explanations.

Hammond, P and Sergot, MJ, 1983. "A PROLOG shell for logic based expert systems" *Proc. 3rd BCS Expert Systems Conf.* Cambridge,95–104. Description of APES, Augmented Prolog for Expert Systems. APES builds upon Prolog execution and representation by offering a user interface that includes HOW and WHY explanations, and Query-the-User, wherein the user is treated as a data-base, and can be queried for missing information.

Hasling, DW, Clancey, WJ and Rennels, G, 1984. "Strategic explanations for a diagnostic consultation system" *International Journal of Man-Machine studies* **20** 3–19. Hasling's main thesis is that to be useful, an expert system must be able to explain its problem solving strategies. the NEOMYCIN knowledge base includes meta-level knowledge in the form of tasks (metalevel goals and subgoals) and metarules (methods for performing tasks). This information is used in making clear the plans and methods used in reaching a goal. The general approach to problem solving is mentioned, as well as the particular action taken. The user can examine the strategy, using normal How and Why questions to explore the meta-rule space.

Hollan, JD, Hutchins, EL and Weitzman, L, 1984. "STEAMER: an interactive inspectable simulation-based training system" *AI Magazine* **5**(2) 15–27. Description of the STEAMER system.

Joshi, A and Webber, B, 1984. "Living up to expectations: computing expert responses" *Proc. AAAI-84,* Austin, Texas, 169–175. In cooperative man–machine interactions, it is necessary but not sufficient for a system to respond truthfully to a user's question. If the system believes that a planned response might mislead the user, it must block that conclusion by modifying its response. This paper characterizes tractable cases in which the system can anticipate the possibility of the user drawing false conclusions, and develops a formal method for computing the inferences that a user might draw from a response from the system.

Kosy, DW and Wise, BP, 1984. "Self-explanatory financial planning models" *Proc. AAAI-84,* Austin, Texas, 176–181. When explaining the results of a computation presented by a financial modelling system, simply presenting the formulas and the input data is not sufficient. Kosey's strategy is to: distinguish the relevant parts of the model by determining the focus of the question; distinguish the significant effects by finding a set of variables in the model that are applicable; translate quantitative information into qualitative (e.g., X goes up because Y went down); and present the explanation using output templates.

McKeown, KR, Wish, M and Matthews, K, 1985. "Tailoring explanations for the user" *Proc. 9th International Joint Conference on Artificial Intelligence,* Los Angeles, California, 794–798. An expert system should provide explanations that correspond to the concerns of the user. Explanations can be tailored by inferring the user's point of view and goal from a brief discourse segment. Builds on Allen (1980) to derive the user's goal based on a series of utterances (rather than a single one).

Neches, R, Swartout, WR and Moore, JD, 1985. "Enhanced maintenance and explanation of expert systems through explicit models of their development" *IEEE Transactions on Software Engineering* **11** 1337–1351. Identifies some of the shortcomings of XPLAIN and attempts to improve them. Describes EES (Explainable Expert Systems) approach. The knowledge engineer produces a semantic model of the domain, which is used by an automatic program writer. XPLAIN had context-dependent terms, limited

number of question types, and was limited to goal refinement. EES implements a mapping between terms and concepts, expanded question types, and goal reformulation if subgoal refinement fails. Explanations are produced according to question types, each with an associated strategy for answering.

Richer, MH and Clancey WJ, 1985. "GUIDON-WATCH: a graphic interface for viewing a knowledge-based system" *IEEE Computer Graphics and Applications* **5** 51–64.

Rissland, EL, Valcarce, EM and Ashley, KD, 1984. "Explaining and arguing with examples" *Proc. AAAI-84,* Austin, Texas, 288–294. Explores the use of examples in two domains–on-line help and legal reasoning. On-line help can be made more intelligent by embedding examples in explanations. Legal argumentation can be strengthened through the use of hypotheticals. Examples can be customized to fit a user's ability or circumstance, using knowledge about the user's directory and files, for example.

Schlobohm, DA and Waterman, DA, 1987. "Explanation for an expert system that performs estate planning" *Proc. First International Conference on AI and Law,* Boston, Massachusetts. EPS consults with a client to create a will. Since the user is typically unknowledgeable about the domain, the system must explain its actions and educate the user. EPS provides several types of explanations. Definitional explanations are assembled from the frame hierarchy, using pop-up menus. How-concluded, similar to HOW, use justification procedures attached to rules which generate text. EPS will return a list of suggestions which match the user's needs. These are used for Alternative-plan explanations and Compare-and-contrast explanations.

Scott, AC, Clancey, WJ, Davis, R and Shortliffe, EH, 1977. "Explanation capabilities of knowledge-based production systems" in: Buchanan, BG and Shortliffe, EH, eds., *Rule-Based Expert Systems*, Addison-Wesley. Mycin's explanation capability was expanded to be able to answer questions. The Explanation Capability comprises two modules: the RSC (reasoning status checker) is used during consultation to allow the user to examine the reasoning chain (how and why questions). The GQA (general question answerer) uses natural language routines to allow the user to ask questions about the conclusions, the static knowledge base, facts, rules, etc. A set of answering specialists are capable of answering questions on a particular topic, e.g., static knowledge or judgmental knowledge.

Southwick, RW, 1988. "Topic explanation in expert systems" in: Kelly, B and Rector, A, eds., *Research and Development in Expert Systems V* Cambridge University Press. A system can explain its strategy through the selection of 'landmarks' that designate topics in the knowledge base.

Swartout, WR, 1981. "Explaining and justifying expert consulting programs" *Proc. 7th International Joint Conference on Artificial Intelligence* Vancouver, BC, 815–822.

Swartout, WR, 1983. "XPLAIN: a system for creating and explaining expert consulting programs" *Artificial Intelligence* **21** 285–325. Good explanations should be capable of presenting the reasoning and justification behind the actions taken by a program. XPLAIN writes the domain program itself, thereby remembering why it did what it did. Knowledge is separated into a Domain Model, comprising of descriptive facts, and Domain Principles, "how-to" methods and rules. The Writer module starts from a top-level goal, and gradually refines it by creating more specific subgoals, until the level of system primitives is reached. Explanations are produced using knowledge in the domain model, domain principles, and the execution trace.

Wallis, JW and Shortliffe, EH, 1982. "Explanatory power for medical expert systems: studies in the representation for clinical consultations" Stanford Dept of Computer Science Report CS-82-923. The goal of the research is the generation of customized explanations, aimed according to the experience and knowledge of the user. When providing explanations of causal reasoning, leave out the concepts that are not understood. User understanding of concepts can be determined through the use of a user-selected difficulty level. If the causal chain for a query has the form t1 $\Rightarrow$ t2 $\Rightarrow$ t3 and t2 is not understood (according to the difficulty level), the explanation presented would be t1 $\Rightarrow$ t3.

Weiner, JL, 1980. "BLAH, a system which explains its reasoning" *Artificial Intelligence* **15** 19–48. BLAH is primarily concerned with structuring explanations so that they do not appear too complex. It uses a TMS, where each rule has justifications for belief attached to it. The knowledge base is segmented into a system and user's view, allowing system to reason using one set of information. Explanations can then be generated using the other, so that details already known by the user can be deleted. The knowledge base is also split into partititions which contain rules that are related in some way. Explanations are assembled, using natural language templates, from a node in a reasoning tree in such a way that the underlying structure of the reasoning tree can be recovered from the explanation.

Theoretical papers on explanation

Allen, JF and Perrault, CR, 1980. "Analyzing intention in utterances" *Artificial Intelligence* **15** 143–178. Describes a model of cooperative behaviour and describes how such a model can be applied in a natural

language understanding system. Discusses several types of speech acts, and the formulation of actions and plans to deal with them. Develops goal inference techniques using plausible inference rules, representation of domain plans, and representation of speech plans.

Cawsey, A, 1989. "Explanatory dialogues" *Interacting with Computers* **1** 69–92. Adopts a grammar-based approach for explanation construction.

Draper, SW, 1987. "Explanation, paradox and abduction" *Proc. 2nd Workshop of the Explanation Special Interest Group* 10–14, Alvey Knowledge Based Systems Club. If external explanations are absent, people will construct their own explanations when forming abductive hypotheses, and when dealing with paradoxes. In this context, the process of explanation-seeking is driven by a need to fit data into preexisting models, but is more of an active constructive process.

Draper, SW, 1987. "A user-centred concept of explanation" *Proc. 2nd Workshop of the Explanation Special Interest Group* 15–23, Alvey Knowledge Based Systems Club. There is little linguistic marking of the role of an utterance, so it is difficult to determine what kind of explanation is desired by a user. Consequently, taxonomies of question types have little promise. Explanations must calculate the difference between the Explainer's and the Inquirer's belief sets, and must recognize the Inquirer's intention in asking. This last will usually require an extended dialogue.

Goguen, JA, Weiner, JL and Linde, C, 1983. "Reasoning and natural explanation" *International Journal of Man-Machine Studies* **19** 521–559. Presents a precise and computationally effective model of the structure of human explanation. Natural explanations are represented in a tree structure whose nodes correspond to the three major types of justification: giving a reason, giving an example, and eliminating alternatives. Explanation production is represented by a sequence of transformations on the tree. Focus of attention is represented by pointers in the tree, and shifts of focus by pointer movement.

Gumpertz, JJ and Hymes, D, 1972. *Directions in Sociolinguistics: The Ethnography of Communication* Holt, Rinehart and Winston. Ethnomethodology.

Hempel, CG, 1965. *Aspects of Scientific Explanation* Free Press. Philosophical investigations into scientific or deductive theories of explanation.

Hughes, S, 1986. "Question classification in rule-based systems" *Proc. Expert Systems '86, British Computer Society Specialist Group on Expert Systems.* An implementation of Lehnert's "question type" model for rule-based systems. Current expert systems are able to treat few of the 14 question types identified by Hughes.

Joshi, A, Webber, B and Sag, I (eds.), 1981. *Elements of Discourse Understanding* Cambridge University Press. Collection of essays on discourse understanding and processing.

Lewis, C and Mack, RL, 1982. *The role of abduction in learning to use computer systems* Human Factors Research Report RC 9433, IBM. Experiments in determining how people develop theories about computer operation.

Linde, C and Goguen, JA, 1978. "Structure of planning discourse" *Journal of Social Biological Structure* **1** 219–251. Using Watergate transcripts, this paper studies the structure of planning discourse. Planning is a mode of discourse with a regular structure, and a cooperatively formulated plan can be represented as a tree structure.

Papers on knowledge representation

Buchanan, BG and Shortliffe, EH (eds.), 1984. *Rule-based Expert Systems* Addison-Wesley. Collection of articles, previously published, about the Stanford Heuristic Programming project MYCIN.

Clancey, WJ and Letsinger, R, 1981. "NEOMYCIN: reconfiguring a rule-based expert system for applications to teaching" *Proc. the 7th International Joint Conference on Artificial Intelligence,* Vancouver B.C., Canada, 829–836. Describes attempts to reuse the MYCIN rule base for a tutoring system.

Davis, R and Buchanan, BG, 1977. "Meta-level knowledge: overview and applications" *Proc. 5th International Joint Conference on Artificial Intelligence* MIT, Massachusetts, 819–826.

de Kleer, J and Brown, JS, 1985. "A qualitative physics based on confluences" in: JR Hobbs and RC Moore (eds.), *Formal Theories of the Common-Sense World* 109–183, Ablex. Develops a theory of qualitative causal physics to describe the behaviour of systems. System variables do not take quantitative values, but are simply assigned one of the qualitative values +, − or 0. A confluence is a qualitative differential equation, used as a modelling tool. Using this qualitative physics, many of the concepts of classical physics are derived.

Klein, D and Finin, T, 1987. "What's in a deep model?" *Proc. IJCAI-87,* Milan, Italy, 559–562. Gives an operational definition of 'knowledge depth' intended to be useful to knowledge engineers in characterizing deep and shallow models. A definition of the relation *deeper-than* states that one model is deeper than another if there is implicit knowledge in the second that is explicit in the first.

AI techniques useful in explanation

Bobrow, DG and Stefik, M, 1983. *The LOOPS Manual* Xerox Corporation.

Chandrasekaran, AB and Mittal, S, 1983. "Deep versus compiled knowledge approaches to diagnostic problem-solving" *International Journal of Man–Machine Studies* **19** 425–436. The authors discuss the relationship between deep and compiled knowledge in expert systems. They claim that most extant systems employ rules that are simply pattern-decision pairs. They refute the claim that deep knowledge is necessary for reasoning, when a compiled version (D) of that knowledge can handle all problems that a deep-knowledge system (U) could. If D fails to solve a case in some way, it is due to one of several reasons: the information is missing in U; D's problem-solving strategy is too weak; D is improperly compiled from U; the compilation process causes a combinational explosion, the reduction of which results in a loss of completeness. Satisfactory explanations can be generated from D; if further (deeper) explanations are required, then a text string summing up the knowledge in U can be added to each node in D. Deep knowledge is not necessarily causal in nature, as some have argued.

Coombs, M and Alty, J, 1984. "Expert systems: an alternative paradigm" *International Journal of Man–Machine Studies* **20** 21–43. Human experts are most often called upon to assist other experts in extending and refining their understanding of a problem at the junction of two domains of knowledge. The first section of the paper describes human interaction in the domain of computer advice. The strategy favoured by participants involved the generation and then critiquing of explanations for some set of problem phenomena. The MINDPAD system was implemented to aid novice Prolog programmers. The programmer enters a problem (in the form of a Prolog program), then an explanation. The system checks the user's idea of how the program will execute against its own, and then tells the user what is wrong, so the user can supply a new explanation.

de Kleer, J, 1986. "An assumption-based truth maintenance system" *Artificial Intelligence* **28** 127–162. Introduces a truth maintenance system that records the base assumptions that support some datum. The ATMS operates in a breadth-first manner, eliminating all backtracking, and permitting reasoning with several possibly inconsistent contexts.

Josephson, JR, Chandrasekaran, B, Smith, JW and Tanner, MC, 1987. "A mechanism for forming composite explanatory hypotheses" *IEEE Transactions on Systems, Man and Cybernetics* **17**(3) 445–454. In order to perform "abductive inference" (going from data to an explanatory hypothesis), a mechanism is presented that assembles hypothesis parts into a unified explanatory hypothesis. The criteria for "best" are internal consistency, explanatory power, plausibility, consistency with the evidence, and parsimony. The assembler uses the data to be explained, a set of sub-hypotheses, and a plausibility rating to select the best explanation.

Langlotz, CP and Shortliffe, EH, 1983. "Adapting a consultation system to critique user plans" *International Journal of Man–Machine Studies* **19** 479–496. ONCOCIN (cancer therapy) is adapted to accept, analyse and critique a physician's plan; to explain the significant differences between the system's plan and the user's. Data are entered through the "Interviewer", while the "Reasoner" uses a rule-based reasoning system to arrive at a recommendation. The user enters his plan, and the system employs hierarchical plan analysis to determine where the two plans differ. Because the domain is hierarchical, it is possible to find the most general set of differences. Explanations of the difference set are produced using an agenda of parameters that differ. The user may select an item from the agenda to be explained, and if in that explanation further parameters are encountered, they are added to the agenda.

Lehnert, WG, 1978. *The Process of Question Answering* Lawrence Erlbaum Associates. Uses a taxonomy of question types to drive a question answering facility. The TEXT system operates in a conceptual dependency context, splitting a question into its query part and CD concept.

McDonald, DD, 1982. "Natural language generation as a computational problem, an introduction" in: Brady, M, ed., *Computational Models of Discourse* MIT Press. Describes Mumble, a full-size text generation system.

McKeown, KR, 1985. *Text Generation* Cambridge University Press. The use of discourse strategies and focus constraints to generate natural language text.

Miller, PL, 1984. *A Critique Approach to Expert Computer Advice: ATTENDING* Pitman. Presents an alternative paradigm to the standard MYCIN-style expert system. In Miller's approach, the user presents his idea of a solution, and the system critiques it, using its expert knowledge base. Miller uses an ATN to represent knowledge in his domain, anesthetic management. Given a user-entered plan, the system can produce possible alternatives, weed out the high-risk options, and produce a comparison between the user's plan and the system's. Polished text is produced by attaching template parts to nodes in the ATN, so that text is built up as the network is traversed.

Rich, EA, 1979. "User modeling via stereotypes" *Cognitive Science* **3** 329–354. User models may be built by using known user characteristics to select a "stereotype" model that partially fits the user, and then individualizing that model to match user details.

Rich, EA, 1983. "Users are individuals: individualizing user models" *International Journal of Man–Machine Studies* **18** 199–214. This paper is concerned with building individual, implicit, long-term models. Techniques are discussed: identification of concepts used; gauging responses that satisfy the user; and using stereotypes to generate many facts from few.

Rissland, EL, 1983. "Examples in legal reasoning" *Proc. 8th International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany.

Self, JA, 1977. "Concept teaching" *Artificial Intelligence* **9** 197–221. Illustration of some design principles for concept teaching in CAI. Program and human concept learning performance is compared, and the incorporation of a concept learning program into a teaching system is discussed.

Sergot, MJ, 1983. "A query-the-user facility of logic programming" in: Degano, P and Sandewall, E, eds., *Integrated Interactive Computer Systems* 27–41, North-Holland. A model of the user as a logical database is presented. This model is useful in expert systems that request missing information from a user.

Shortliffe, EH, 1976. *Computer-based medical consultations: MYCIN* Elsevier. The MYCIN handbook.

Sleeman, D, 1984. "UMFE: a user modelling front end subsystem" Stanford Research Report. UMFE determines the user's level of sophistication by asking a few questions, then presents an answer to a question in terms of concepts understood by the user. The knowledge base includes a list of domain concepts, each with a difficulty and importance rating. The system sets the difficulty level by interactively asking the user whether a concept is understood. If it is, it is assumed that all of its siblings with the same importance value are also known. See Wallis (1982) for initial work that this paper builds on.

Southwick, RW, 1990. *A reason maintenance system for backward reasoning systems* Research report, DOC 90/11, Imperial College, London. Describes reason maintenance techniques for backward reasoning systems, in order to eliminate redundant processing and maintain a consistent set of beliefs.