Report 77-01
Stanford -- KSL

Explanation Capabilities of Production-Based
Consultation Systems. A. Carlisle Scott,
Randall Davis, William J. Clancey,
Edward H. Shortliffe, 1977

card 1 of 1

EXPLANATION CAPABILITIES OF PRODUCTION-BASED CONSULTATION SYSTEMS

STAN-77-593
Heuristic Programming Project Memo 77-1

A. Carlisle Scott, Randall Davis, William Clancey,
and Edward H. Shortliffe
Knowledge-Based Consultation Systems Group

ABSTRACT

A computer program that models an expert in a given domain is more likely to be accepted by experts in that domain, and by non-experts seeking its advice, if the system can explain its actions. An explanation capability not only adds to the system's credibility, but also enables the non-expert user to learn from it. Furthermore, clear explanations allow an expert to check the system's "reasoning", possibly discovering the need for refinements and additions to the system's knowledge base. In a developing system, an explanation capability can be used as a debugging aid to verify that additions to the system are working as they should.

This paper discusses the general characteristics of explanation systems: what types of explanations they should be able to give, what types of knowledge will be needed in order to give these explanations, and how this knowledge might be organized. The explanation facility in MYCIN is discussed as an illustration of how the various problems might be approached.

KEY WORDS

PRODUCTION RULES, COMPUTER-BASED CONSULTANTS, KNOWLEDGE-BASED SYSTEMS, EXPLANATION, QUESTION-ANSWERING, JUDGMENTAL KNOWLEDGE, NATURAL LANGUAGE UNDERSTANDING

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency, the Bureau of Health Sciences Research and Evaluation, or the United States Government.

This research was support in part by the Bureau of Health Sciences Research and Evaluation under Grant HS-01544, and by the Defense Advanced Research Projects Agency under ARPA Order No.2494, Contract No. DAHC 15-73-C-0435.

Table of Contents

Section Page

Subsection

1 General Discussion

1.1 Consultative Production Systems

A consultation program plays the role of an expert consultant in some domain, giving advice or answers to non-experts with problems in the domain. Users will often want to know how the system arrived at its results during a particular consultation. This paper explains how the implementation of such a program as a production system can facilitate program-generated explanations.

A production system [2] consists of three basic components: a set of production rules, a data base which is both used and updated by these rules, and a rule interpreter. A production rule often is in the form of a situation-action rule: it describes a situation and a set of actions to be taken if this situation is found to exist. The rule interpreter determines the order in which rules will be tried, checks to see if the situations exist, and undertakes the required actions. It also determines how many of the potentially useful rules will be used: only the first (where ordering may be predetermined or computed dynamically), all possible rules, or enough rules to satisfy some criterion that the interpreter uses.

In some production systems, rules are always tried in a predetermined order. In others, the order in which rules are tried varies with different consultations, since a rule will be tried as soon as the rule interpreter determines that it may be useful. In such systems, the common alternatives are data-directed rule invocation, in which a rule is considered "useful" if its situation part matches the data base, and goal-directed rule invocation, in which a rule is "useful" if its action part will help the system reach its current goal. Many systems use a combination of goal- and data-directed rule invocation.

A consultative production system need not be a psychological model, imitating a human's reasoning process. The important point is that the system and a human expert use the same (or similar) knowledge about the domain to arrive at the same answer to a given problem. The system's rules and data base can be viewed as a *knowledge base* containing the domain-specific knowledge of an expert as well as facts about a particular problem. When a rule is used, its actions make changes to the data base which are the system's *decisions* or *deductions*. Thus, a rule can be thought of as a piece of *judgmental knowledge*, using the judgment and knowledge of an expert to make deductions.

The process of trying rules and taking actions can be thought of as "reasoning", and explanations consist of showing how rules used information provided by the user to make various intermediate deductions and finally to arrive at the answer. If the information contained in these rules is sufficient to show why an action was taken (without getting into programming details), an explanation can consist of printing each rule that was used (or an English equivalent of what the rule means.)

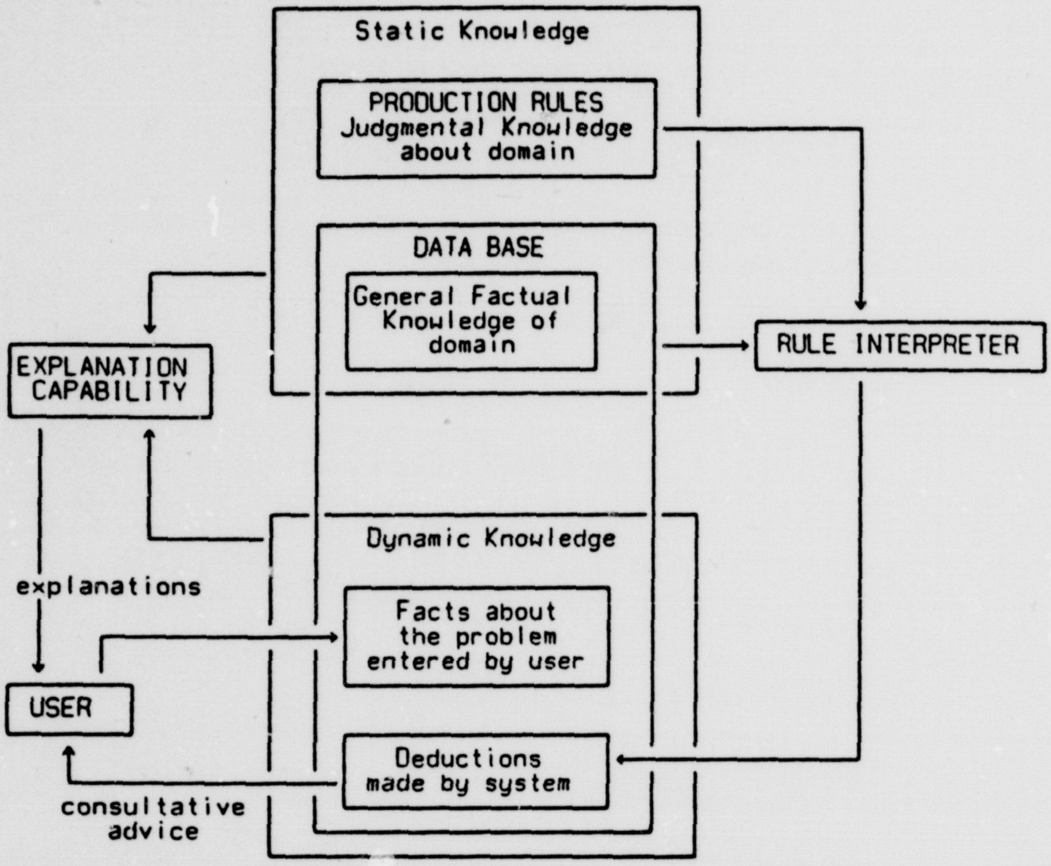
Static Knowledge
PRODUCTION RULES
Judgmental Knowledge about domain
DATA BASE
General Factual Knowledge of domain
RULE INTERPRETER
EXPLANATION CAPABILITY
Dynamic Knowledge
explanations
Facts about the problem entered by user
USER
Deductions made by system
consultative advice

Figure 1. A Production-Based Consultation System wit'ı Explanation Capability

The three components of a production system (a RULE INTERPRETER, a set of PRODUCTION RULES, and a DATA BASE) are augmented by an EXPLANATION CAPABILITY. The data base is made up of general facts about the system's domain of expertise, facts that the user enters about a specific problem, and deductions made about the problem by the system's rules. These deductions form the basis of the system's consultative advice.

The explanation capability makes use of the system's knowledge base to give the user explanations. This knowledge base is made up of static domain-specific knowledge (both factual and judgmental) and dynamic knowledge specific to a particular problem.

1.2 Performance Characteristics of an Explanation Capability

The purpose of an explanation capability (EC) is to give the user access to as much of the system's knowledge as possible. Ideally, it should be easy for a user to get a complete, understandable answer to any sort of question about the system's knowledge and operation -- both in general, and with reference to a particular consultation. This implies three major goals in the development of an explanation capability:

1) To ensure that the EC can handle questions about all relevant aspects of the system's knowledge and actions. It should be capable of giving a few basic types of explanations, for example:

   How it made a certain decision
   How it used a piece of information
   What decision it made about some subproblem
   Why it didn't use a certain piece of information
   Why it failed to make a certain decision
   Why it required a certain piece of information
   Why it didn't require a certain piece of information
   How it will find out a certain piece of information [while the consultation is in progress]
   What the system is currently doing? [while the consultation is in progress]

   The specific set of explanation types which are chosen as basics, however, will depend on the particular system.

2) To enable the user to get an explanation which answers the question completely and comprehensively.

3) To make the EC easy to use. A novice should be able to use the EC without first spending a large amount of time learning how to request explanations.

We will distinguish two slightly different functions for an EC, and divide it into two components: the reasoning-status checker (RSC) to be used during the consultation, and the general question answerer (GQA) to be used during the consultation or after the system has printed its results.

A reasoning-status checker will answer questions asked during a consultation about the status of the system's reasoning process. A few simple commands are often sufficient to handle the questions that the RSC is expected to answer.

A general question-answer will answer questions about the current state of the system's knowledge base, including both static domain knowledge, and facts accumulated during the consultation. A GQA will often need the ability to recognize a wide range of question types about many aspects of the system's knowledge. For this reason, it might be difficult to define a few simple commands which would be easy to learn and still cover all the possible questions that might be asked. Consequently, natural-language processing in this component may be important to an explanation system's acceptability.

In an interactive consultation, the system periodically requests information about the problem. This offers the user an opportunity to request explanations while the consultation is in progress. In non- interactive consultations, the user has no opportunity to interact with the system until after it has printed its conclusions. Unless there is some mechanism allowing a user to interrupt the reasoning process and ask questions, the explanation capability for such a system will be limited to questions about the system's final knowledge state. It will have no reasoning-status checker, and its general question-answerer will only be accessible at the termination of the consultation.

1.3 Knowledge Requirements of an Explanation Capability

An EC must know what is in the system's knowledge base, and how it is organized. In order to give explanations of the system's current (or previous) actions, an EC also needs to understand how the system's rule interpreter works: when rules will be tried, how they can fail, what causes the interpreter to try one rule but not another, etc. This general "schema" for how or why certain rules are used, together with a comprehensive record of the specific actions taken during a particular consultation, can be used as a basis for explaining the results of that consultation.

A reasoning-status checker will need a record of what the system has done so far in order to explain how it arrived at the current step. General knowledge of how the rule interpreter works is necessary in order to explain where the current step will lead. The ability to understand individual rules also may be necessary to the extent that the content of a rule may explain why it was necessary to use this rule, or may affect which future rules will be tried.

A general question-answerer will need more information about the system since the scope of its explanations is much broader: its task is to answer general questions about the system's knowledge base. To do this, it must know how the system stores knowledge about its area of expertise (the static knowledge with which it starts each consultation) and how it stores facts gathered during a particular consultation (its dynamic knowledge). These two types of information will allow a GQA to answer questions about the substance and extent of the production system's current knowledge.

If an explanation capability also is to provide information about how the system arrived at the facts that are currently in its dynamic knowledge base, the GQA will need all the information that a reasoning-status checker uses: a detailed record of the consultation, an understanding of the rule interpreter, and the ability to understand rules.

These three types of knowledge could be supplemented with a limited amount of general information about such things as elementary logic, set theory, and arithmetic comparisons. This would allow the GQA to answer more complicated questions about why the system's knowledge base is in its current state, and to answer questions involving relationships between different facts in the knowledge base.

The nature of the consultation domain, as well as what primary purpose the explanation capability is to serve, will influence the range of questions that an EC should handle. In some systems, a simple retrieval of facts may suffice, while others may need to give detailed description of the production system's "decision" process and to make a number of deductions from facts that it has.