Hierarchically Structured Production Rules*

Philip Leith
Faculty of Mathematics, Open University, Milton Keynes, UK

Production rules when used in 'Expert Systems'[1] have shown themselves to be effective means of encapsulating an expert's 'knowledge' about how to solve a problem. The ELI (Expert Legislative Information) system uses a novel means of hierarchically structuring these production rules which: (i) allows the production rules to be accessed in a parallel manner; (ii) allows the production rules to self-control access; (iii) allows each condition in the production to represent a richer semantic body of information than traditional production rule systems. In this paper, I discuss the notion of knowledge engineering and why hierarchically structured production rules can be useful.

INTRODUCTION

One of the current areas of research in computer science is the area known as 'knowledge engineering', which has been described as:

> . . . the art of bringing the principles and tools of AI research to bear on difficult applications problems requiring expert's knowledge for their solution. The technical issues of acquiring this knowledge, representing it, and using it appropriately to construct and explain lines-of-reasoning, are important problems in the design of knowledge-based systems.[2]

One of the fundamental techniques of knowledge engineering is the removal from the programmed statements of the knowledge which the programmed system has about the problem domain. Thus the most simplistic model of such a system allows two modules (an *interpreter* and a *knowledge base*) intrinsic to the system as described in Fig. 1.

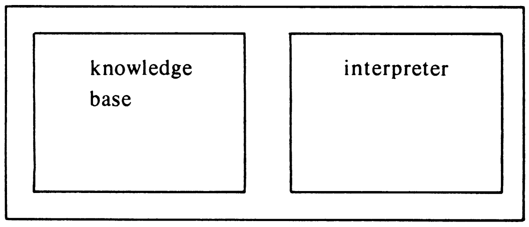

Figure 1. Model of a knowledge engineered system.

The knowledge base is that part of the system which contains information about problem solving techniques. It is simply a data structure which cannot itself produce any advice unless it is interpreted by the interpreter.

The knowledge engineering approach is not the only way in which computer programs can be produced to provide high quality advice or operate in certain domains; rather it can be considered that due to certain advantages inherent in the approach it is the better way of producing programs for certain areas. Three of the advantages which accrue from the knowledge engineering approach are:

(i) Since the problem solving knowledge is held separate from that part of the program which uses the knowledge, in areas where the knowledge is constantly changing only the knowledge (held in a data base known as the 'Knowledge Base' (KB)) about the problem area needs be changed—the program which uses that knowledge will theoretically never need to be revised after it is initially debugged (although adding extra facilities to the system will necessitate this). The ELI system is an example of a system which benefits from this approach because it operates in the field of legislation, an area where new law is constantly being passed and added to the statute book and case law decisions provide new interpretations of that law.

(ii) Ease of incorporating an individual's own knowledge. This of course, follows from (i) above. An individual who wishes to program a computer to carry out certain tasks in an area that the individual knows well, has a ready produced program with which he can interact and which will accept his specific expertise without requiring him to be conversant with the programming language or the workings of the system.

(iii) The type of knowledge which can be represented by the system and the ease with which that knowledge can be represented can be experimented with. This aspect takes account of the fact that 'knowledge' is not a concrete entity; it is not easily describable and is not tangible. In fact the subject is so intangible that most AI researchers have avoided providing a definition of the knowledge which their particular system handles.

In this paper, I describe one element (the knowledge base) of an expert system which uses a novel means of structuring the productions it contains. The techniques I have used can be classified as those of the knowledge engineering field.

PRODUCTION RULES

Production rules (PRs) or productions are representations of certain situation/action 'chunks of knowledge' (I shall

* The production rule handling section of ELI, discussed in this paper, is available (along with basic documentation) from the author on receipt of a small magnetic tape.

CCC–0010–4620/83/0026–0001 $02.50

simply cyclically define a 'chunk of knowledge' as either a production rule, or the information which can be gleaned from one or more conditions of that production rule)—they show what action should occur if such a situation is in effect. The situation is represented by a pattern of conditions which appear to the left of the PR. The action is represented by a goal to the right of the PR. A production rule of the form:

⟨condition1⟩ ⟨condition2⟩ ⟨condition3⟩ ⇒ ⟨goal*x*⟩

states that 'if a state exists that the three conditions specified as condition1, condition2 and condition3 are found to be true, then the action specified by goal*x* must be put into effect.'

The generality of the conditions and goals which can be stated is proven by the various areas in which PRs have been used (see Ref. 3 for examples). But one of their most important features is that they seem to accord well with human means of representing certain states and consequent actions which should be put into effect; and that they provide a means whereby experts within a certain domain can represent knowledge about their own particular domain.

This simple model of production above outlines the general format of production rules; it does not however deal with the format of the individual conditions, perhaps the most problematic design aspect of production rule systems. MYCIN[4] used a format for each condition of the form:

⟨predicate function⟩ ⟨object⟩ ⟨attribute⟩ ⟨value⟩

This allowed automatic generation of LISP code from constrained English, and also allows the use of meta-knowledge.[5,6] Davis *et al.*[7] noted that although the format of MYCIN's stylized rules was essentially non-restrictive within the domain, there was a tendency towards some rules with awkwardly long and complicated premises, and creation of desired triggering patterns was a sometimes non-trivial task.

The choice of rule format for ELI was constrained by the fact that there seemed to be no suitable stylization which could cope with rules such as:*

```
IF (entitled to supplementary benefit)
   (claimant is a prisoner)
   (claimant is on remand)
   (period of detention is less than one year)
   (claimant responsible for own accommodation)
   (accommodation will be lost through debt)
THEN
   (single debt payment available)
```

The decision was made to use pattern matching techniques. One method currently being used in the process of integrating new rules into the KB, is the application of a function (the system is programmed in INTERLISP) to each new condition to remove 'noise' words (such as TO, IS, CLAIMANT etc.) from the input condition. Depending on the length of the resulting condition, only conditions/goals which also contain a proportion (about a third) of the remaining words from the already assimilated condition are recognized as possible matches by the system. The system is not left to decide upon difficult (or, for that matter 'easy') conceptual matches. The problems of just which conditions might 'match' are illustrated by the two phrases in Fig. 2.

(went to office on monday)

and

(went to office on monday afternoon)

Figure 2. An example of a 'difficult' conceptual match.

Only by having access to contextual information about both the conditions (i.e. the causal chain of which each are an element, and the goals they might lead to) can the user decide whether they are in fact a conceptual match. The user, in ELI, is therefore presented with the two patterns (and the relevant rules can be presented along with any notations attached to individual conditions) and must decide whether the two conditions are identical.

This aspect, the conceptual matching of conditions, is one which I intend to spend future research time on, perhaps incorporating lexical analysis of the input rules. My research is currently also moving into the area of producing information about the chains of reasoning provided by the system itself (which as Ref. 8 states is essential to a legal representative) rather than the 'simple' production of advice, and such matching seems to be an essential part of such research.

ELI's KNOWLEDGE BASE

A conceptual diagram of the knowledge base can be seen in Fig. 3.

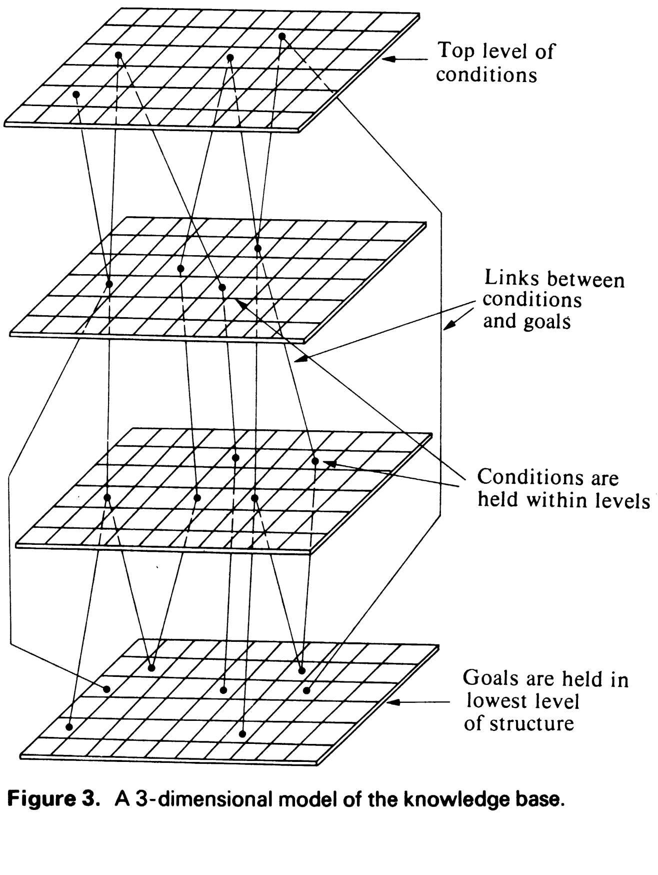
Top level of conditions
Links between conditions and goals
Conditions are held within levels
Goals are held in lowest level of structure

Figure 3. A 3-dimensional model of the knowledge base.

* The individual conditions of each rule could be more complex than this, but for several reasons (e.g. ease of collecting knowledge from expert, the architecture of the knowledge base and the attempt—dealt with later—to provide a basis upon which reasoning can be built) it was decided to keep the conditions as short as possible.

The basic facets of the KB are:

(i) Individual conditions can be physically shared by rules

One immediate advantage of this approach is that in a situation where one condition is common to many PRs, a substantial amount of storage (and hence search space) can be saved. But this is not the most important aspect; rather it is that conditions can be treated as the most important elementary object. Held in this manner they are examinable in three ways:

(a) as individual conditions which represent one chunk of causal knowledge
(b) as parts of an individual rule they represent one cause for that rule having (or not having) triggered
(c) as common elements of more than one rule, they represent common aspects of the pattern of triggering of those rules.

These three reasons are particularly important in the ELI domain where an attempt is made to provide a semblance of legal reasoning; however, this is beyond the scope of this paper, since the notion of legal reasoning is a much more specific and idiosyncratic subject than other reasoning.

(ii) Causal links are from condition to following condition to eventual goal

It is this aspect which allows to the PRs the ability of self control of access. A two-dimensional representation (nodes representing conditions and arcs representing links) in Fig. 4 illustrates this.

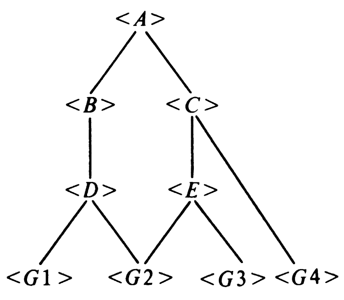

Figure 4.

In a traditional (i.e. vertically listed) KB, these PRs would be held in the form illustrated by Fig. 5.*

$$
\begin{aligned}
&\langle A\rangle\langle B\rangle\langle D\rangle \Rightarrow \langle G1\rangle\\
&\langle A\rangle\langle B\rangle\langle D\rangle \Rightarrow \langle G2\rangle\\
&\langle A\rangle\langle C\rangle\langle E\rangle \Rightarrow \langle G2\rangle\\
&\langle A\rangle\langle C\rangle\langle E\rangle \Rightarrow \langle G3\rangle\\
&\langle A\rangle\langle C\rangle \qquad\;\; \Rightarrow \langle G4\rangle
\end{aligned}
$$

Figure 5.

* Or they could be contracted by a 'knowledge expert' into a smaller number of more conceptually complex PRs, e.g.:

$$\langle A\rangle\langle B\rangle\langle D\rangle \Rightarrow \langle G1\rangle\langle G2\rangle$$

However, as the author has stated,[9] it is this intrusion of knowledge expert which arrests the direct interaction of expert with system.

Thus four individual tests of condition $\langle A\rangle$ (more if $\langle A\rangle$ is not the first condition in each PR) are required in the traditional production system; only one is required in the ELI KB.

Davis[6] noted the problem of 'saturation', a state he describes as

> the situation in which so many applicable knowledge sources are retrieved that it is unrealistic to consider exhaustive, unguided invocation.

and he proposed meta-control to guide the intepreter towards a selective set of proposals. To a certain extent, the process of structuring PRs hierarchically within a KB as done in ELI, allows an amount of meta-control; thus the explicit presentation of the top level of the ELI KB selectively controls access to one or more implicitly held individual rules (e.g. $\langle B\rangle\langle D\rangle\langle G1\rangle$ can be considered as an implicitly held rule which is accessed only when the meta-condition $\langle A\rangle$ is true). By reducing the number of initial tests needed to access these implicit rules, the chances of saturation are reduced.

(iii) One 'top level' of conditions is continually presented to the interpreter

It is this aspect which allows parallel access to the PRs. In a traditional system, a cyclic (and serial) search pattern is operated by initially testing the first production in the list, then proceeding down the list testing each production until a goal is triggered. When a goal is triggered the data base of known facts is updated, and the interpreter then returns to the top condition in the list and retests through the upper productions again. This cyclic pattern is continued until either the desired goal is effected, or no more productions can be fired (i.e. the interpreter has searched from top to bottom of the list without having triggered any goal).

The traditional pattern, apart from problems of saturation mentioned above, is a particularly inflexible means of searching (see Ref. 10 for further information on flexibility in procedural control). It does not allow the user the opportunity to stop the search at any point and (while keeping in mind the current position in the search) try a search which might be more important to the user at that particular time. Also it debars the user from changing the forward search motion to a goal to condition search (i.e. backtracking search). The usefulness of this can be seen in the example shown in Fig. 6.

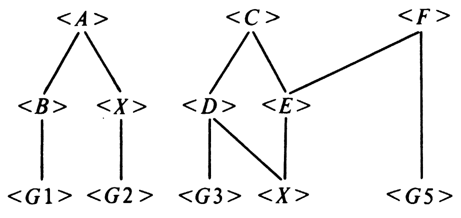

Figure 6.

Thus in ELI, $\langle A\rangle$ is first tested and if found true (and there are several means of assessing the truth of the condition—one means is by examining notations attached to each condition—see Ref. 11 for a rationale), then ⟨B⟩ is next tested. If ⟨B⟩ is found to be false, ⟨X⟩ is tested. The system can halt at this point (either under its own control or the user's) and search the action base to find whether there is an incidence of this condition also being a goal. If there is a goal ⟨X⟩ (as above) then the system can backtrack ⟨X⟩ to ⟨D⟩ and ⟨C⟩; or ⟨X⟩ to ⟨E⟩ and ⟨C⟩; or ⟨X⟩ to ⟨E⟩ and ⟨F⟩; until ⟨X⟩ is either proved true or false. If ⟨X⟩ is found to be true then the database of known facts can be updated and the interpreter can return to ⟨X⟩ below ⟨A⟩ and continue its search pattern.

This is particularly useful as a strategy within ELI's legislative domain; since the users of the system can be expected to be knowledgeable (to some extent at least) of the legislation contained in the system they can be encouraged by the system to interact with it using their own knowledge of the legislation. Using this approach adds to the description of the system as an 'intelligent tool'.

Another aspect of this presentation of a top level of conditions is that modules of knowledge can be easily added to or deleted from the system. Thus blocks or modules of PRs can be added which have little or no relationship with the PRs in other modules (but do aid in the presentation of improved advice to the end user) as illustrated in Fig. 7.

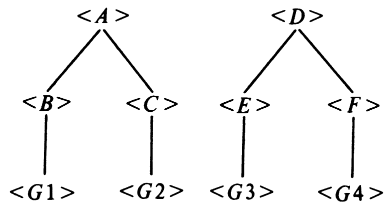

Figure 7.

The addition of the module below ⟨D⟩ had no effect upon the module below ⟨A⟩ because they have no common goals. The deletion of ⟨D⟩ cannot therefore affect triggering of rules in the module below ⟨A⟩ (this may well affect the quality or type of advice being produced however). This might be useful in some problem domains, but it is not particularly useful in a system which deals with legislation, which by its nature cannot be simply segmented into, say, separate Acts of Parliament—so much of law (even after consolidating acts have been passed) transcends the idea of simple non-interacting blocks of knowledge.

(iv) There may be any number of levels of conditions

The only deciding factor for the number of levels in the fully built up KB is the 'depth' or complexity of the productions. The KB automatically expands to accept any number of levels, from 1 to n (although whether large numbers of levels are useful in most domains is another question—in most situations it seems more rational to break up rules, so that intermediate goals can be created).

(v) The goals are held in one separate plane

This allows the easy access of backtracking procedures (as in (iii) above) to the KB, which in effect provides definitions of concepts; thus

(entitled to supplementary benefit)

can be defined by the causal pattern of conditions held above it.

METHODS OF INCORPORATING RULES

Since this paper deals with the handling of PRs within the KB, I shall omit description of the preparation of rules for input to the system, suffice to say that the test set for the ELI system were prepared using copies of the relevant acts and a guide to these acts.

When the system kernel described in this paper is presented with a fully formed rule, how is it incorporated?

There are three main techniques, which are applied to the rule in the following order: (a) each of the input conditions is matched against the top level conditions until a match between an input and an already assimilated condition is found (if not found, then procedure (b) is followed). The rule number of the input rule is then attached to this condition and the links to lower conditions are retrieved. The same matching technique is then used on the conditions one link below the top level condition. This process continues until no match can be found; at which point the remaining conditions from the input rule are then inserted by themselves. The goal base is then tested to find a match for the input goal; if one is found then the input goal is assimilated with that goal, else the input goal is simply appended to the goal list. The general pattern of integration from this method is illustrated in Fig. 8.

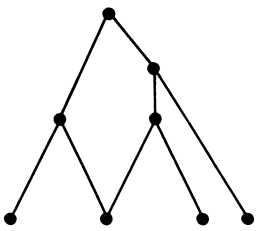

Figure 8. Top down incorporation.

(b) If procedure (a) is ineffective (i.e. no match with a top level condition can be found) then an attempt is made to match the input goal with an already assimilated goal. If no match is found then procedure (c) is followed. If a match has been found, however, then the process of (a) is attempted in reverse, i.e. trying to associate the conditions with already assimilated conditions from lower levels upwards. If a point comes when no further matches are found, then the remaining conditions are inserted by themselves, one being placed on the top level. The pattern of integration is illustrated in Fig. 9.

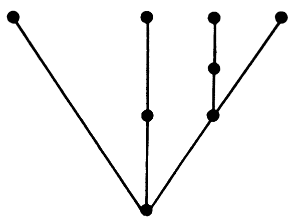

Figure 9. Goal up incorporation.

(c) If (b) has not been successful (i.e. no matching goal can be found), then the conditions are inserted as entire rules; i.e. one condition is placed on the top level, the goal is placed in the goal list, and the remaining conditions are placed in the successive levels above the goal. The pattern is illustrated in Fig. 10.

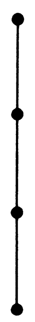

Figure 10. Rule incorporated 'alone'.

Links between the conditions are held as addresses on the property list of each condition; the rule numbers to which each condition belongs are also held on the property lists of each condition.

REASONING BASIS

Within the ELI system, the basis for the reasoning chain is the individual condition; it provides the semantic basis upon which the user can decide the truth of the condition. The conditions themselves frequently provide insufficient information for the user requirements; for example, in a consultation with a client, a welfare rights worker must make reference as to whether a particular condition held in the system has been extracted from the Supplementary Benefit Act, a tribunal judgement or the procedure followed by a Supplementary Benefit Officer—this information is essential in the preparation of an appeal to a Supplementary Benefit Tribunal.

Within the ELI system I have attempted to broaden the amount of information which can be gleaned from each condition so that, for example, the system can tell the user where a condition was extracted from by the expert.

Generally, I describe each condition as a semantic token, upon which other information can be attached so that the user is provided with more information than would otherwise be possible. Included within this idea of information which can be attached to a semantic token are the notations which can be associated both with production rules and individual conditions themselves. Thus each piece of reasoning chain can be associated with a specific piece of precedent (e.g. from the Law Reports), or a section of legislation.

Thus the attempt has been made to present as rich a source of semantic information to the user as possible; both to help to judge the truth of each condition, and also so that extra information can be extracted from the system by the end user.

SUMMARY

Unlike previous means of handling production rules, this system does not store the PRs in one (or more) lists. The advantages of this kind of structure are:

(i) Parallel access can be achieved to the productions.
(ii) Each condition on the upper level can control access to several productions below.
(iii) Backtracking from one goal can allow access to several productions above.
(iv) The above three items, together with the ability to notate conditions and rules, allow a rich semantic quality to be given to each element within the knowledge base.

REFERENCES

1. D. Michie, *Expert Systems in the Micro-Electronic Age,* Edinburgh University Press, Edinburgh (1979).
2. E. A. Feigenbaum, Themes and case studies of knowledge engineering, in *Proceedings International Joint Conference on Artificial Intelligence—77* (1977).
3. M. Stefik *et al.* The organization of expert systems, a tutorial, *Artificial Intelligence,* Vol. 18 (1982).
4. E. H. Shortliffe, Mycin: A Rule-Based Computer Program for Advising Physicians Regarding Antimicrobial Therapy Selection, Doctoral Thesis, Stanford University (1975).
5. R. Davis, Content reference: reasoning about rules, *Artificial Intelligence,* Vol. 15 (1980).
6. R. Davis, Meta-rules: reasoning about control, *Artificial Intelligence* **15** (1980).
7. R. Davis *et al.* Production rules as a representation for a knowledge-based consultation program, *Artificial Intelligence* Vol. 8 (1977).
8. N. MacCormick, *Legal Reasoning and Legal Theory,* Oxford University Press, Oxford (1979).
9. P. Leith, ELI: an expert legislative consultant, IEE Conference on Man/Machine Systems (1982).
10. M. P. Georgeff, Procedural control in production systems, *Artificial Intelligence,* Vol. 18 (1982).
11. I. P. Goldstein and E. Grimson, Annotated production systems, a model for skill acquisition, *Proceedings International Joint Conference on Artificial Intelligence—77* (1977).

Received August 1981