Report 79-07
Stanford -- KSL

Scientific DataLink

A Domain-Independent Production-Rule System for Consultation Programs.
William van Melle,
Aug 1979

card 1 of 1

HPP-79-7

V-305

A domain-independent production-rule system for consultation programs.

William van Melle
Heuristic Programming Project
Department of Computer Science
Stanford University
Stanford, California 94305

Abstract

EMYCIN is a programming system for writing knowledge-based consultation programs with a production-rule representation of knowledge. Several major components of the system, including an explanation program and knowledge acquisition routines, are described. EMYCIN has been used to build consultation systems in several areas of medicine, as well as an engineering domain. These experiences lead to some general conclusions regarding the potential applicability of EMYCIN to new domains.

Keywords: knowledge-based systems, production rules, knowledge representation, automated consultant.

This work was supported in part by the NSF grant and the Advanced Research Projects Agency, contract .

1 Introduction

The focus of much current work in artificial intelligence is the development of computer programs that aid scientists with complex reasoning tasks. Recent work has suggested that one key to the creation of intelligent systems is the incorporation of large amounts of task-specific knowledge. Building such programs from scratch can be a very time-consuming task. We have found that the basic framework of the MYCIN program [Shortliffe76] could be generalized sufficiently to make possible the substitution of knowledge bases from other domains. The result of this generalization is EMYCIN, [1] a system for developing rule-based consultation programs. The aim of this work is to make knowledge-based systems technology more accessible to investigators in many fields.

2 Background

Some of the earliest work in artificial intelligence centered around attempts to create generalized problem solvers. Work on programs like GPS [Newell72] and theorem proving [Nilsson71], for instance, was inspired by the apparent generality of human intelligence and motivated by the belief that it might prove possible to develop a single program applicable to many problems. While this early work demonstrated that there was a large body of useful general purpose techniques (such as problem decomposition into subgoals and heuristic search in its many forms), these techniques alone did not offer sufficient power for high performance in complex domains.

Recent work has instead focused on the incorporation of large amounts of task-specific knowledge in what have been called "knowledge-based" systems. Rather than non-specific problem solving power, knowledge-based systems have emphasized high performance based on the accumulation of large amounts of knowledge about a single domain. Some examples to

[1] "Essential MYCIN", i.e. MYCIN stripped of its domain knowledge. EMYCIN is written in Interlisp and runs on a DEC PDP-10.

date include efforts at symbolic manipulation of algebraic expressions [MACSYMA74], speech understanding [Lesser74], chemical inference [Buchanan78], and medical consultants [Pople77], [Shortliffe76].

While these systems display an encouraging level of performance, each is powerful in only a very narrow domain, and assembling the knowledge base and constructing a working program for such domains is a difficult, continuous task that has often extended over several years. However, one such system, the MYCIN program for infectious disease consultation, included in its design the goal of keeping the domain knowledge well separated from the program which manipulates that knowledge. Thus, while MYCIN itself is specific to the domain of infectious diseases and thus not at all general, the basic rule methodology employed and the facilities that support it could provide a foundation for a general rule-based system. This basic framework has now been extracted and is embodied in the EMYCIN system. EMYCIN shares with GPS the philosophy of separating the deductive mechanism from the problem-specific knowledge; however, EMYCIN's extensive user facilities make it much more accessible than the early systems. Our hope is that with such a system it will be possible for workers in other domains to rapidly build functional consultation programs.

3 Description of EMYCIN

EMYCIN is used to construct a *consultation program*, by which we mean a program which offers its user advice on some problem within its domain of expertise, much as a human consultant does. The program elicits information relevant to the case by asking the user questions during the course of the consultation. The program then applies its knowledge to the facts of the case and informs the user of its conclusions. The user is free to ask the program questions about its reasoning in order to better understand or validate the advice given.

3.1 Knowledge Organization

3.1.1 Facts

Knowledge in EMYCIN is represented in terms of production rules operating on associative (attribute-object-value) triples. The *objects* are actual or conceptual entities in the domain of the consultation. Various *attributes* characterize the objects. Questions asked during the consultation attempt to fill in the *values* for relevant attributes of the objects.

To represent the uncertainty of data or competing hypotheses, attached to each triple is a *certainty factor* (CF), a number between -1 and 1 indicating the strength of the belief in that fact. A CF of 1 represents total certainty, while a CF of -1 represents certainty in the negation of the fact. While certainty factors are not conditional probabilities, they are informally based in probability theory (see [Shortliffe75] for details).

Typical triples from some of EMYCIN's application domains are

```
(a)  GEOMETRY       of SUB-STRUCTURE-2 is PLANAR (1.0)
(b)  TIME-DEPENDENT of STRUCTURE-24    is YES (-1.0)
(c)  IDENTITY       of ORGANISM-1      is PSEUDOMONAS (.8)
(d)  IDENTITY       of ORGANISM-1      is E.COLI (.15);
```

in other words, (a) the geometry of the second sub-structure is planar, (b) Structure 24 has *no* time-dependent terms in its equilibrium equations, (c) Organism 1 is probably Pseudomonas, but (d) there is some evidence to believe it is E. coli.

The system reasons about its domain using knowledge encoded as *production rules* [Davis77]. Each rule has a *premise*, which is a conjunction of predicates over triples in the knowledge base. If the premise is true, the conclusion in the *action* part of the rule is drawn. If the premise is known with less than certainty, the strength of the conclusion is modified accordingly (see [Shortliffe75]). Figure 1 shows typical rules from the domains of infectious diseases and structural analysis (described in Section 4).

Rule 35

```
If:   1) the gram stain of the organism is gram negative, and
      2) the morphology of the organism is rod, and
      3) the aerobicity of the organism is anaerobic,
then:  There is suggestive evidence (.6) that the identity
         of the organism is Bacteroides.

PREMISE:  ($AND (SAME CNTXT GRAM GRAMNEG)
                (SAME CNTXT MORPH ROD)
                (SAME CNTXT AIR ANAEROBIC))

ACTION:   (CONCLUDE CNTXT IDENT BACTEROIDES TALLY .6)
```

Rule 68

```
If:  1) The material composing the sub-structure is one of
        the metals,
     2) The analysis error (in percent) that is tolerable
        is less than 5,
     3) The non-dimensional stress of the sub-structure is
        greater than .5, and
     4) The number of cycles the loading is to be applied
        is greater than 10000
Then:  It is definite (1.0) that fatigue is one of the
          stress behaviour phenomena in the sub-structure

PREMISE:  ($AND (SAME CNTXT COMPOSITION (LISTOF METALS))
                (LESSP* (VAL1 CNTXT ERROR)
                        5)
                (GREATERP* (VAL1 CNTXT ND-STRESS)
                           .5)
                (GREATERP* (VAL1 CNTXT CYCLES)
                           10000))
ACTION:  (CONCLUDE CNTXT SS-STRESS FATIGUE TALLY 1.0)
```

Figure 1. Sample rules from two EMYCIN domains.

The predicates are simple Lisp functions operating on associative triples. $AND, the multivalued analogue of the Boolean AND function, performs a minimization operation on CF's. The body of the rule is actually an executable piece of Lisp code, and "evaluating" a rule entails little more than the Lisp function EVAL.

3.2 Application of Rules -- the Rule Interpreter

The control structure is a goal-directed backward-chaining of rules. At any given time, EMYCIN is working toward establishing the value of some attribute. To this end, the system retrieves the (precomputed) list of rules whose conclusions bear on this goal. Rule 68 above, for example, would be retrieved in the attempt to determine the stress of a substructure. If, in the course of evaluating the premise of one of these rules, some other piece of information is needed which is not yet known, EMYCIN sets up a subgoal to find out that information; this in turn causes other rules to be tried. Questions are asked during the consultation when the rules fail to deduce the necessary information.[2] If the user cannot supply the requested information, it remains unknown, and rules that require it will simply fail. Thus, the rules unwind to produce a succession of goals and it is the attempt to achieve each goal that drives the consultation.[3]

Note that this control structure is able to deal gracefully with incomplete information. If the user is unable to supply some piece of data, the rules which use that data simply fail; if the remaining rules are unable to make conclusions, or if there simply aren't any rules for some particular goal, the system might make a weaker conclusion (lower CF) or fail to make the conclusion altogether, and end up asking the user. Thus, the expert can construct and test the rule base in pieces; one part can be tested, with the user (expert) supplying the answers to goals for which no rules have been written yet.

[2] Attributes may instead be flagged as information likely to be immediately available from the user, in which case the system asks first, and only tries rules if the user cannot supply the information.

[3] EMYCIN also permits a limited use of *antecedent rules*, which may be triggered when the program receives new data or makes a conclusion. They are essentially demons waiting for their premises to become true, and hence never cause questions to be asked.

3.3 More on the rule representation

Many of the system's important features are facilitated by the use of rules as the primary representation of knowledge. Since each rule is intended to be a single "chunk" of information, the knowledge base is inherently modular, making it relatively easy to update. Individual rules can be added, deleted or modified without drastically affecting the overall performance of the system. The rules are also a convenient unit for explanation purposes.

The simple, stylized nature of the rules is also useful. The English translation of a Lisp rule, for example, is made possible by the association of a translation pattern with each predicate function, indicating the logical roles of the function's arguments.

While the syntax of rules permits the use of any Lisp functions, there is a small set of standard predicates which make up the vast majority of the rules. The system contains information about the use of these predicates in the form of function *templates*. For example, the predicate SAME is described as follows:

```
function template:     (SAME OBJECT ATTR VALUE)
sample function call:  (SAME CNTXT  SITE BLOOD)
```

The system can use these templates to "read" its own rules. For example, the template shown here contains the standard tokens OBJECT, ATTR and VALUE, indicating the components of the associative triple which SAME tests. If the clause above appears in the premise of a given rule, the system can determine that the rule needs to know the site of the culture, and in particular that it tests whether the culture site is blood. When asked to display rules which are relevant to blood cultures, the system will be able to select that rule.

The basic predicates supplied in EMYCIN include ones that test whether an attribute of an object is (is not, might be, definitely is) some value, and others which perform comparisons and computations on numeric-valued attributes (such as a patient's age). The system builder can extend the basic set by writing new functions (in Lisp) and supplying appropriate templates (for rule reading) and English translations. We have found that new domains typically need a small set of new predicates tailored to the domain. For example, the PUFF system (Section 4) includes a predicate for comparing disease severities expressed as tokens "MILD", "MODERATE" and "SEVERE".

3.4 Explanation Capability

EMYCIN's *explanation program* allows the user to interrogate the system's knowledge, either to find out about inferences made in the particular case at hand or to examine the static knowledge base in general. During the consultation, EMYCIN can offer explanations of the current, past and likely future lines of reasoning. As described above, the consultation program requests various pieces of information. If the motivation for any particular question is unclear, the user may temporarily put off answering and instead inquire why the information is needed. Since each question is asked during the attempt to evaluate some rule, a first approximation at an answer is simply to display the rule currently under consideration. The program can also explain what reasoning led to the current point, and what use might later be made of the information being requested. This is possible by examining traces left by the rule interpreter, and by reading the rules in the knowledge base to determine which are applicable. This form of explanation requires no language understanding on the part of the program; it is invoked by simple commands from the user ("WHY" and "HOW"). Figure 2 shows some examples of this type of explanation.

A more general form of explanation is available via the *QA Module*, which is automatically invoked after the consultation has ended, and can also be entered during the consultation to answer questions other than the special WHY/HOW described above. The QA Module accepts simple English-language questions dealing with any conclusion drawn during the consultation, or about the domain in general (Figure 3). The level of explanation is again that of the rules, presumably comprehensible to anyone familiar with the domain, even if not familiar with computers in general or the intricacies of the EMYCIN system. The questions are parsed by pattern matching and keyword lookup, using a dictionary which is automatically constructed from the phrases used in defining the various objects and attributes of the domain. The QA module is described in more detail elsewhere [Scott77].

```
    34) Was penicillinase added to this blood culture (CULTURE-1)?
    ** WHY
[i.e. WHY is it important to determine whether penicillinase was added
to CULTURE-1?]

[3.0] This will aid in determining whether ORGANISM-1 is a contaminant.
  It has already been established that
     [3.1] the site of CULTURE-1 is blood, and
     [3.2] the gram stain of ORGANISM-1 is grampos
  Therefore, if
     [3.3] penicillinase was added to this blood culture
  then
     there is weakly suggestive evidence (.4) that ORGANISM-1 is a
         contaminant
[RULE039]

    ** WHY
[i.e. WHY is it important to determine whether ORGANISM-1 is a contaminant?]

[4.0] ... in order to determine whether there is significant disease
associated with this occurrence of ORGANISM-1.
  It has already been established that
     [4.1] this blood culture was taken from a sterile source
  Therefore, if
     [4.2] ORGANISM-1 is not a contaminant
  then
     there is strongly suggestive evidence (.9) that there is
     significant disease associated with this occurrence of ORGANISM-1
[RULE032]

    ** WHY
[i.e. WHY is it important to determine whether there is significant
     disease associated with this occurrence of ORGANISM-1?]

[5.0] ... in order to determine the organisms which require therapy.
  It has already been established that
     [5.1] the infection is known, and
     [5.2] the identity of ORGANISM-1 is known
  Therefore, if
     [5.3] there is significant disease associated with this
         occurrence of ORGANISM-1
  then
     ORGANISM-1 is one of the organisms which require therapy
[RULE090]

    ** HOW 5.2
[i.e. HOW was it established that the identity of ORGANISM-1 is known?]

Question 9 was asked, but you weren't sure; so the following was used:

  [5.5] RULE084 was used to conclude that the identity of ORGANISM-1
        is Streptococcus-group-a (.29).
```

Figure 2. Examples of WHY and HOW questions in the infectious disease domain. (Users' questions follow a double asterisk; everything else is typed by the system.)

```
** WHAT IS THE GEOMETRY OF SUB-STRUCTURE 1
The geometry of the floor section (SUB-STRUCTURE-1) is planar

** HOW DO YOU USE THE GEOMETRY OF A SUB-STRUCTURE
The following rules use:
     the geometry of the sub-structure

53, 48, 102, 101, 100

The parameters concluded by these rules are:
     the shape of the sub-structure
     the analysis recommendations to be considered when preparing
          the structure for modelling

Which of the rules do you wish to see?
      ** 53

RULE053

  If:  1) Buckling is one of the deflection phenomena in the
          structure, and
       2) You have examined the sub-structures, and have found one
          for which the geometry of this sub-structure is curved
  Then:  It is definite (1.0) that the following is one of the
           analysis recommendations to be considered when
           preparing the structure for modelling: Buckling
           estimates should be based on incremental load
           analysis.

                              ---------------

** IS ORGANISM-1 A STREPTOCOCCUS?

No.  The identity of ORGANISM-1 is E.coli (.74) Bacteroides (.7)
    Klebsiella-pneumoniae (.4) Pseudomonas-aeruginosa (.22)
    Enterobacter (.21) Proteus-non-mirabilis (.19) Serratia (.03)
    Salmonella (.02)

** WHY DID YOU RULE OUT STREPTOCOCCUS AS A POSSIBILITY FOR ORGANISM-1?

RULE033 could have been used to determine that the identity of
    ORGANISM-1 is Streptococcus.  However, clause 2 of rule033 ["the
    morphology of the organism is coccus"] was already known to be false
    for ORGANISM-1, so the rule was never tried.

** IS BACTEROIDES AEROBIC?

No, the aerobicity of Bacteroides is anaerobic.
```

Figure 3. Examples of the QA Module from two EMYCIN domains.

The explanation program is also a helpful high-level debugging aid for the system developer. Without having to resort to Lisp-level manipulations, she can examine any inferences that were made, find out why others failed, and thereby correct errors or omissions in the knowledge base. The Teiresias program [Davis76] developed the system's WHY and HOW capability for just such a task: to guide the expert through the program's reasoning chain and locate faulty (or missing) rules.[4]

3.5 Knowledge Acquisition

The initial step in building an EMYCIN consultation program is to construct the knowledge base. The knowledge consists of the rules and the object-attribute structures upon which they operate. Since a substantial part of the initial effort involved in creating a new system is spent in the acquisition phase, we have tried to provide programmatic assistance for this task.

The knowledge base must eventually reside in the internal Lisp format which the system uses to run the consultation, answer questions and the like. At the very basic level, one could imagine a programmer using the Lisp editor to create the necessary data structures totally by hand; here the entire translation from the expert's conceptual rule to Lisp data structures is performed by the programmer. At the other extreme, the expert would enter rules in simple English, with the entire burden of understanding placed on the program.

The actual choice in EMYCIN represents a point between these extremes. Entering rules at the base Lisp level is fraught with tedium and errors; understanding English rules is far too difficult, especially in a new domain where the vocabulary hasn't even been identified and organized for the program's use (just recognizing new attributes in free English text is a major obstacle). EMYCIN instead supplies a high-level database editor for the data

[4] Teiresias is not currently available in EMYCIN, but we are exploring the possibility of adding it as a system debugger or English front end to the rule acquisition routines.

structures in the system. The data must still be entered in its Lisp format, but the editor handles all the bookkeeping involved in managing the rule base and performs various checks to catch a number of common user errors.

3.5.1 Description of the associative triples

The *objects* in a consultation are organized into what we term the *context tree*. An object, or *context*, is simply one of the nodes in the tree. Generally there are multiple instances of each type of context other than the root. A context tree from a typical infectious disease consultation is shown below.

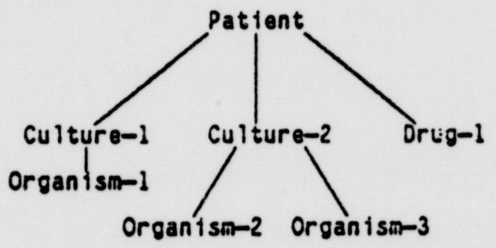

Figure 4. Sample context tree.

The context tree serves two main purposes. First, it indicates the explicit relationships among objects in the consultation, providing a sort of "binding" mechanism: one rule can refer to attributes of more than one context, e.g. the site of a culture and the identity of the organism growing out of the culture, and the system will make the correct associations. Second, it structures the consultation. EMYCIN operates by setting up the root of the tree and from there setting up depth-first the other nodes of the tree, inquiring about basic information at each node (e.g. age of the patient, site and date of the culture). The consultation thus is more structured than it would be if questions were ordered only by the system's logical requirements for information.

The system builder supplies both the context types and their interconnections. Each context is described by its position in the tree, an English translation of what it represents, and an indication of which attributes belong to it. Some simple domains have no context tree at all, in which case only the root of the degenerate tree (i.e. the central object of the consultation) need be described. In this case, all of the system attributes apply to the root context.

Attributes are simply any properties which may be used to describe a context. The system builder describes each attribute by giving it an English translation (SITE: "the site of the culture"), indicating which context type(s) it can belong to, and what values it may take on.

3.5.2 Rule entry

The rules are stored as Lisp forms, a conjunction of predicates for the *premise* and one or more conclusions for the *action*. To partially bridge the gap between precise Lisp input and free-form English text, there is an intermediate "terse" rule format. This format is a simplified Algol-like language which uses the attributes and their values as operands; the operators correspond to EMYCIN predicates. E.g. SACON's Rule 68 shown in Figure 1 above could be written instead as:

```
If  Composition = (LISTOF METALS) and
    Error < 5 and
    Nd-stress > .5 and
    Cycles > 10000
Then   Ss-stress = fatigue
```

Figure 5. Example of stylized English format for rule input.

The terse rule format is also available on output, for users who prefer it to the more verbose English translations.

As each rule is entered, it is checked for syntactic validity, to catch spelling errors and other simple errors. The check is made by comparing each clause with its template, and seeing that, for example, each ATTR slot is filled by a valid attribute, and that its VALU slot is a legal value for the attribute. Other data structures are also updated, which tell the rule interpreter which rules conclude about which attribute.

3.6 Other support features

EMYCIN has facilities for maintaining a library of sample cases. These can be used for testing a complete system, or for debugging a growing one. The user's answers to all the questions asked during the consultation are simply stored away, indexed by their object and attribute. When a library case is rerun, the answers are looked up and automatically supplied; any new questions resulting from changes in the rule base are asked in the normal fashion. This makes it easy to check the performance of a new set of rules on a "standard" case.

There is also a *Batch* program which permits the user to run any or all cases in the library in background mode; the system reports back whether any changes in the "results" of the consultation occurred, and invokes the QA module to explain why they occurred. The system builder need only indicate to the system which attributes represent the "results" or important intermediate steps of the consultation. Thus the user can easily catch cases where a change to the rule base caused previously successful cases to fail. This could be viewed as a form of semantic checking to supplement the syntactic checking routinely performed at the time of rule acquisition.

EMYCIN's existing human-engineering features relieve the system builder of many of the tedious cosmetic concerns of producing a human-useable program. Most of these center around the consultation interaction where the system requests data from the user:

1. The explanation program, which lets the user know why a question is being asked, has already been described.

2. The terminal input facility includes such features as spelling correction and

tenex-style completion on altmode from a list of possible answers (typically the list of legal values for an attribute as supplied by the system designer).

3. EMYCIN provides simple help in the form of what the legal answers to a question are, should the user be confused. The system designer can also include more substantial help by giving rephrasings or elaborations on the original question; these are simply entered via the database editor as additional properties of the attribute in question.

4 Applications

Several consultation systems have been written in EMYCIN. The original MYCIN program (now implemented in EMYCIN) provides advice on the diagnosis of and therapy for patients with infectious diseases. There are currently some 450 rules in the MYCIN rule set, with 13 types of contexts and 235 attributes; the rules make conclusions about 85 of the attributes, with the remaining attributes simply being input data. Results of formal evaluations of MYCIN's competence in the domains of bacteremia (bacterial infections in the blood) and meningitis indicate that MYCIN's performance in these areas has begun to approach that of medical specialists [Yu79].

The PUFF system [Kunz78] performs interpretation of measurements from the pulmonary function laboratory. The project is a collaboration of a pulmonary physiologist, biomedical engineers and Stanford computer scientists who had previous experience with the MYCIN program. The data from over 100 cases were used to create some 60 production rules diagnosing the presence of pulmonary disease. A sample PUFF rule is shown below. These rules are used to create a complete report including the input measurements, historical information, and the measurement interpretation.

```
If:  1) The degree of obstructive airways disease of the
        patient is greater than or equal to mild,
     2) The degree of diffusion defect of the patient is
        greater than or equal to mild, and
     3) The total lung capacity measured by the body box
        (TLCB) is greater than 110 percent of predicted,
Then: 1) There is strongly suggestive evidence (.9) that
         the subtype of obstructive airways disease is
         emphysema, and
      2) The following is one of the conclusion statements
         about this interpretation: "The low diffusing
         capacity, in combination with obstruction and a
         high total lung capacity would be consistent with
         a diagnosis of emphysema."
```

The HEADMED program [Heiser78] is an application of EMYCIN to clinical psychopharmacology. The system diagnoses a range of psychiatric disorders and can recommend drug treatment if indicated. The system contains 275 rules at present, one of which is shown below.

```
If:  1) The diagnosis for which therapy is being
        recommended is major-depressive-disorder,
     2) The intensity of this patient's psychiatric
        disturbance is one of: moderate, mild,
     3) The current mental state of the patient is
        psychotic, and
     4) There is no recent prior episode of
        psychiatric disturbance
Then:  There is strongly suggestive evidence (.8) that the
          class or catagory of treatments is antipsychotic
```

As a stronger test of domain independence, EMYCIN was applied to the completely non-medical domain of structural analysis. SACON (Structural Analysis Consultation) [Bennett78] provides advice to a structural engineer regarding the use of a large structural analysis program called MARC. The MARC program uses finite-element analysis techniques to simulate the mechanical behavior of objects. The engineer typically knows what she wants the MARC program to do, e.g. examine the behavior of a specific structure under expected loading conditions, but does not know how the simulation program should be set up to do it. The goal of the SACON program is to recommend an analysis strategy; this advice can then be used to direct the MARC user in the choice of specific input data, numerical methods and material properties.

The performance of the SACON program matches that of a human consultant for the limited domain of structural analysis problems that was initially selected. To bring the SACON program to its present level of performance, about two man-months of the experts' time were required to explicate their task as consultants and formulate the knowledge base, and about the same amount of time to implement and test the rules. The system contains some 160 rules, 4 context types, and 50 attributes, half of which are concluded by rules.

5 Range of Applicability -- Limitations

Although we do not have a precise characterization of domains which are most suitable for an EMYCIN application, our experience provides some general guidelines. We found that many of the simpler systems did not even make use of the inexact inference mechanism provided by CFs. It is possible that more efficient representations exist for the precise knowledge existing in such domains. However, it still seems appropriate to build a reasoning program from EMYCIN as a first test of the inference rules written by an expert. While many of the system's complicated features, such as certainty factors and the context tree, may go unused in the simpler systems, those features do not substantially burden a program which does not use their extra generality.

However, even in the domains that have been successfully implemented so far, it has not always been easy to actually formulate the domain knowledge into production rules. Our desire to keep the rule format simple is occasionally at odds with the need to encode the many aspects of scientific decision-making. For example, the process of therapy selection in MYCIN proved more amenable to a generate and test algorithm [Clancey77]. The backward-chaining of rules by the deductive system is also often a stumbling block for experts who are new to the system; however, they soon learn to structure their knowledge appropriately. In fact, some experts have felt that encoding their knowledge into rules has helped them formalize their own view of the domain, leading to greater consistency in their decisions.

The representation of facts as associative triples with CF's may not be natural or adequate for many domains. However, the triples have proved a flexible enough data structure in our investigations thus far.[5].

The specific control structure in EMYCIN, viz., goal-directed backward chaining of rules, might be considered much more restrictive. Certainly other control structures and representations are possible; some current efforts, such as KRL [Bobrow77], provide a large array of knowledge-representation choices, allowing the user great flexibility in expressing the knowledge. We recognize that EMYCIN is not as general, nor is it intended to be:[6] there may well be advantages in restricting the user's options. At least during the initial formulation of the knowledge of a domain, the simple framework of a single, easily understood control structure allows the expert to focus on the knowledge in the rules themselves, and not get sidetracked by a bewildering array of control choices.

6 Conclusion

Currently the system builder still requires a fair knowledge of Lisp and the workings of EMYCIN itself. Our work aims at improving the facilities for acquiring and debugging rules, as these are the most time-consuming portion of creating a new consultation system. The next level of improvement is to raise the level of the dialog to the point where a more natural English-like interaction could occur with someone not at all versed in Lisp.

We are continuing to develop EMYCIN toward being a truly convenient "workshop" for knowledge-based system exploration. Our aim is to have a system in which a knowledge engineer can relatively quickly create a functional consultation program in a new domain, try out a new set of rules, or simply discover whether a rule-based system is reasonable for the task.

---

[5] While it would require considerable reprogramming, the triples could be extended or be replaced by a more powerful representation scheme. For example, another researcher at Stanford is adding a time dimension to the stored data and inferences [Fagan78].

[6] Nor does EMYCIN attempt in any way to mimic human reasoning, as production systems such as PSG do.

References

[Bobrow77]
Bobrow, D.G., and Winograd, T. Experience with KRL-0: One Cycle of a knowledge representation language, *Proceedings of IJCAI5*, 1977.

[Bennett78]
Bennett, J.S., *et al.* SACON: A Knowledge-based Consultant for Structural Analysis, Department of Computer Science, Stanford University, Memo HPP-78-23, Sept. 1978.

[Buchanan78]
Buchanan, B.G., and Feigenbaum, E.A., DENDRAL and Meta-DENDRAL: Their Applications Dimension, *Artificial Intelligence* 11:1, 1978.

[Clancey77]
Clancey, W.J. An antibiotic therapy selector which provides for explanations, *Proceedings of IJCAI5*, 1977.

[Davis76]
Davis, R. Applications of Meta-Level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases. Doctoral dissertation, Stanford University, June 1976.

[Davis77]
Davis, R. and King, J. An overview of production systems. *Machine Intelligence 8: Machine Representations of Knowledge* (eds. E.W. Elcock and D. Michie), John Wylie, 1977.

[Fagan78]
Fagan, L. Ventilator Manager: A Program to Provide On-Line Consultative Advice In The Intensive Care Unit. Department of Computer Science, Stanford University, Memo HPP-78-16, September 1978

[Heiser78]
Heiser, J.F., Brooks, R.E., Ballard, J.P., Progress Report: A Computerized Psychopharmacology Advisor, *Proceedings of the 11th Collegium Internationale Neuro-Psychopharmacologicum*. Vienna, 1978.

[Kunz78]
Kunz J.C., *et al.* A physiological rule based system for interpreting pulmonary function test results, Heuristic Programming Project, Computer Science Department, Stanford University, November 1978.

[Lesser74]
Lesser V R, *et al.* Organization of the HEARSAY II speech understanding system, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-23, February 1975, pp 11-23.

[MACSYMA74]
The MACSYMA reference manual, The MATHLAB Group, MIT, September 1974.

[Newell72]
Newell A and Simon H, *Human Problem Solving*, Prentice-Hall, 1972.

[Nilsson71]
Nilsson N J, *Problem Solving Methods in Artificial Intelligence*, McGraw Hill, 1971.

[Pople77]
Pople H, The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning, *Proceedings of IJCAI5*, p. 1030, 1977.

[Scott77]
Scott, A.C., *et al*. Explanation Capabilities Of Knowledge-Based Production Systems. *American Journal of Computational Linguistics*, Microfiche 62, 1977.

[Shortliffe75]
Shortliffe E H, and Buchanan B G. A model of inexact reasoning in medicine, *Mathematical Biosciences* 23, pp351-379, 1975.

[Shortliffe76]
Shortliffe E H, *Computer-based clinical therapeutics: MYCIN*, American Elsevier, 1976.

[Yu79]
Yu, V.L., *et al*, Evaluating the performance of a computer-based consultant. Computer Programs in Biomedicine, 9, 95-102, 1979.