

A DOMAIN-INDEPENDENT PRODUCTION-RULE SYSTEM FOR CONSULTATION PROGRAMS

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

EMYCIN is a programming system for writing knowledge-based consultation programs with a production-rule representation of knowledge. The major components of the system, including an explanation program and knowledge acquisition routines, are briefly 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.

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. However, building such "knowledge-based" programs from scratch can be a very time-consuming task. We have found that the basic framework of the MYCIN program [9] could be generalized sufficiently to make possible the substitution of knowledge bases from other domains. The result of this generalization is EMYCIN*, 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 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.

2.1 Knowledge Organization

Knowledge in EMYCIN is represented in terms of production rules operating on associative (attribute-object-value) triples. 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 [8].

Reasoning is performed using domain knowledge encoded as *production rules*. 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 [8]). Figure 1 shows a typical rule from the domain of structural analysis (described in Section 3).

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

English and internal Lisp forms of a SACON rule.

2.2 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. The rule in Figure 1, 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

* "Essential MYCIN", i.e. MYCIN stripped of its domain knowledge. This work was supported in part by the NSF grant MCS 77-02712 and the Advanced Research Projects Agency, contract MDA 903-77-C-0322.

necessary information. 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 (EMYCIN also permits a limited use of *antecedent rules*, triggered when the program receives new data or makes a conclusion).

2.3 Usefulness of the rule representation

As has been described in earlier reports of the MYCIN program [4], 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 a convenient unit for explanation purposes. The system's ability to "read" its own rules is used extensively by the explanation program (below) and other routines which assist the user in manipulating the rule base.

2.4 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, in response to certain user commands, EMYCIN can offer explanations of the current, past and likely future lines of reasoning (*WHY* a question was asked, or *HOW* a conclusion is reached). 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 outside the scope of the special *WHY/HOW* commands. The *QA Module* accepts simple English-language questions dealing with any conclusion drawn during the consultation, or about the domain in general. 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* has been described in great detail elsewhere [7].

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.

2.5 Knowledge Acquisition

A substantial part of the initial effort involved in creating an EMYCIN system is spent in the acquisition phase, viz. constructing the knowledge base of the rules and the object-attribute structures upon which they operate. Thus we have devoted much effort to providing programmatic assistance for this task.

EMYCIN supplies a high-level database editor for the data structures in the system. The data must be entered in its Lisp format (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), 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.

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 2. 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. Other data structures are also updated, such as the one telling the rule interpreter which rules conclude about which attributes.

2.6 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. When a library consultation is rerun, the system's questions are answered by supplying the response, if any, given when the case was first run. Many such cases may be automatically rerun in background mode, with the system comparing the "results" with those of earlier runs. This makes it easy to check the performance of a new set of rules on "standard" cases.

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; they include a terminal input facility with such features as spelling correction and Tenex-style completion on altmode from a list of possible answers, and a simple help facility.

3 Applications

Several consultation systems have been written in EMYCIN. The MYCIN program (which could be viewed as the original implementation of 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, making conclusions about 85 of the 235 attributes (the remaining attributes are simply 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 [10].

The PUFF system [6] performs interpretation of measurements from the pulmonary function laboratory. The data from over 100 cases were used to create some 60 production rules diagnosing the presence of pulmonary disease. These rules are used to create a complete report including the input measurements, historical information, and the measurement interpretation.

The HEADMED program [5] 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 in Figure 3.

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 category of treatments is antipsychotic

Figure 3. A sample HEADMED rule.

As a stronger test of domain independence, EMYCIN was applied to the completely non-medical domain of structural analysis. SACON (Structural Analysis Consultation) [2] 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 goal of the SACON program is to recommend an analysis strategy to guide the MARC user in the choice of specific input data, numerical methods and material properties. The system contains some 160 rules and 50 attributes, half of which are concluded by the rules.

4 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, may go unused in the simpler systems, those features do not substantially burden a program which does not use their extra generality.

It is not always easy to actually formulate the domain knowledge into production rules, even in the domains that have been successfully implemented so far. 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 [3]. 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. 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. We recognize that EMYCIN is not as general as some other efforts, e.g. KRL [1], nor is it intended to be: 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.

5 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. Our aim is to have a truly convenient "workshop" for knowledge-based system exploration, 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.

Acknowledgements

The author wishes to acknowledge the contribution of Carli Scott, who programmed much of the QA Module and continues to assist in the development of new features in EMYCIN. Thanks to Carli and also to Bruce Buchanan for their helpful advice on this paper.

References

- [1] Bobrow, D.G., and Winograd, T. An overview of KRL-0, a knowledge representation language. *Cognitive Science*, 1,1, 1977.
- [2] Bennett, J.S., et al. SACON: A Knowledge-based Consultant for Structural Analysis, Computer Science Dept, Stanford University, Memo HPP-78-23, Sept. 1978.
- [3] Clancey, W.J. An antibiotic therapy selector which provides for explanations, *Proceedings of IJCAI5*, 1977.
- [4] Davis, R., Buchanan, B.G., and Shortliffe, E.H. Production rules as a representation for a knowledge-based consultation system. *Artificial Intelligence*, 8,15-45, 1977.
- [5] Helser, J.F., Brooks, R.E., Ballard, J.P. Progress Report: A Computerized Psychopharmacology Advisor, *Proceedings of the 11th Collegium Internationale Neuro-Psychopharmacologicum*. Vienna, 1978.
- [6] Kunz J.C., et al. A physiological rule based system for interpreting pulmonary function test results, HPP-78-19 (Working Paper), Heuristic Programming Project, Computer Science Dept, Stanford University, December 1978.
- [7] Scott, A.C., et al. Explanation Capabilities Of Knowledge-Based Production Systems. *American Journal of Computational Linguistics*, Microfiche 62, 1977.
- [8] Shortliffe E. H., and Buchanan B. G. A model of inexact reasoning in medicine, *Mathematical Biosciences* 23, pp351-379, 1975.
- [9] Shortliffe E. H. *Computer-based clinical therapeutics: MYCIN*, American Elsevier, 1976.
- [10] Yu, V.L., et al. Evaluating the performance of a computer-based consultant. *Computer Programs in Biomedicine*, 9, 95-102, 1979.