June 1982

Report No. STAN-G-82-922

An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System

by

Motoi Suwa, A. Carlisle Scott, Edward H. Shortliffe

Department of Computer Science

Stanford University
Stanford, CA 94305

An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System

Motoi Suwa*

A. Carlisle Scott

Edward H. Shortliffe

Heuristic Programming Project

Departments of Computer Science and Medicine

Stanford University

Stanford, California 94305

This work was supported in part by the National Library of Medicine (Program Project Grant LM-03395 and Research Career Development Award LM-00048), the National Science Foundation (Grant MCS-7903753), and the Office of Naval Research (Grant N00014-81-K-0004). It was carried out on the SUMEX-AIM computer (NIH grant RR-00785).

*Present address: . Computer Vision Section
Electrotechnical Laboratory
l-l-4 Umezono, Sakura-mura
Niihari-gun, Ibaraki-ken 305
Japan

Abstract

We describe a program for verifying that a set of rules in an expert system comprehensively spans the knowledge of a specialized domain. The program has been devised and tested within the context of the ONCOCIN System, a rule-based consultant for clinical oncology. The stylized format of ONCOCIN's rules has allowed the automatic detection of a number of common errors as the knowledge base has been developed. This capability suggests a general mechanism for correcting many problems with knowledge base completeness and consistency before they can cause performance errors.

Table of Contents

Acknowledgments

During the development of the program described here, the authors received encouragement and useful suggestions from other members of the ONCOCIM research project. We would like to thank all of those who helped to make the program possible. We are especially grateful to Craig Tovey, Miriam Bischoff, and Bruce Buchanan for numerous valuable comments on earlier versions of this paper.

1 Introduction

The builders of a knowledge-based expert system must ensure that the system will give its users accurate advice or correct solutions to their problems. The process of verifying that a system is accurate and reliable has two distinct canponents: checking that the knowledge base contains all necessary information, and verifying that the program can interpret and apply this information correctly. The first of these components has been the focus of the current research; the second corresponds to the familiar problem of program "debugging" and will not be discussed in this paper.

Knowledge-base debugging, the process of checking that a knowledge base is correct and complete, is one component of the larger problem of knowledge acquisition. This process involves testing and refining the system's knowledge in order to discover and correct a variety of errors that can arise during the process of transferring expertise from a human expert to a computer system. In this paper, we discuss some common problems in knowledge acquisition and debugging, and describe an automated assistant for checking the completeness and consistency of the knowledge base in the ONCOCIN system [3].

2 Knowledge Acquisition

Before knowledge can be embodied in a cunputer system, it must undergo a nunber of transformations. First, a human acquires expertise in some domain through study, research and experience. Next, the expert attempts to formalize this expertise and to express it in the internal representation of an expert system, e.g., production rules, frames, or semantic nets. Finally, the knowledge, in a machine-readable form such as LISP expressions, is added to the computer system's knowledge base.

Problems can arise at any stage in this process: the expert's knowledge may be incomplete, inconsistent, or even partly erroneous. Alternatively, accurate and complete knowledge may not be adequately transferred to the computer-based representation. The latter problem typically occurs when an expert who does not understand computers works with a knowledge engineer who in unfamiliar with the problem domain; misunderstandings that arise are often unrecognized until performance errors occur. Finally, spelling or syntax mistakes that are made when the knowledge-base is entered into the canputer are a frequent source of errors.

3 Why an Automated Assistant for Knowledge-base Debugging?

The knowledge base of an expert system is generally constructed through collaboration between experts in the problem domain and knowledge engineers. The domain experts formulate their knowledge and the knowledge engineers encode this knowledge for use by the system. This difficult and time-consuming task can be facilitated by a program which:

(1) checks for inconsistencies and gaps in the knowledge base,

(2) helps the experts and knowledge engineers to communicate with each other, and

(3) provides a clear and understandable display of the knowledge as the system will use it.

An automated assistant for the system builders could rapidly identify problems in the system's knowledge base and possibly allow the experts to discover gaps in their knowledge or errors in their reasoning.

4 Knowledge-Base Debugging

4.1 Earlier Work

One goal of the TEIRESIAS program [1] was to automate knowledge-base debugging in the context of the MYCIN infectious disease consultation system [2]. TEIRESIAS allowed an expert to judge whether MYCIN's diagnosis was correct, to track down the errors in the knowledge base that led to incorrect conclusions, and to alter, delete or add rules in order to fix these errors. The knowledge transfer occurred in the setting of a problem-solving session; no formal assessment of rules occurred at the time they were initially entered into the knowledge base.

In the EMYCIN system for building knowledge-based consultants [4], the knowledge-acquisition program fixes spelling errors, checks that rules are semantically and syntactically correct, and points out potential erroneous interactions among rules. In addition, EMYCIN's knowledge-base debugging facility includes the following options:

(1) a trace of the system's "reasoning process" during a consultation;

(2) an interactive mechanism for reviewing and correcting the system's conclusions (a generalization of the TEIRESIAS program);

(3) an interface to the system's explanation facility to produce automatically, at the end of a consultation, explanations of how the system reached its results;

(4) a verification mechanism which compares the system's results at the end of a consult with the stored "correct" results for the case that were saved from a previous interaction with the TEIRESIAS-like option. The canparison includes explanations of why the system made its incorrect conclusions and why it did not make the correct ones.

4.2 Systematic Checking of a Knowledge Base

The knowledge-base debugging tools mentioned above allow a system builder to identify problems with the system's knowledge base by observing errors in its performance on test cases. While thorough testing is an essential part of verifying the consistency and canpleteness of a knowledge base, it is rarely possible to guarantee that a knowledge-base is completely debugged , even after hundreds of test runs.

It is not always possible to test a growing knowledge base by running sample cases. TEIRESIAS was developed after the MYCIN system was fully functional and had an extensive rule set. EMYCIN is specifically designed for the incremental growth of a knowledge base by allowing the system builder to run consultations even when only a skeletal knowledge base has been defined. The task of building an EMYCIN system is simply to encode and add the knowledge. In contrast, building a new expert system typically starts with the selection of knowledge representation formalisms and the design of a program to use the knowledge. Only when this had been done it is possible to encode the knowledge and write the program. The system may not be ready to run tests, even on simple cases, until the entire knowledge base is encoded. When an expert system is developed in this manner, it would be convenient if system builders could run a preliminary check on the knowledge base before the full reasoning mechanism is functioning and without gathering actual data for a test run.

Knowledge-base testing tools, therefore, can be augmented by a program which systematically checks a knowledge base for completeness and consistency. This checking can be done during the system's development, even without a fully functioning reasoning mechanism.

4.3 Debugging a Rule-Based System

4.3.1 Logical Checks for Consistency

When knowledge is represented in production rules, inconsistencies in the knowledge base appear as:

CONFLICT: two rules succeed in the same situation but with conflicting results.

REDUNDANCY: two rules succeed in the same situation and have the same results.

SUBSUMPTION: two rules have the same results, but one contains additional restrictions on the situations in which it will succeed. Whenever the more restrictive rule succeeds, the less restrictive rule also succeeds, resulting in redundancy.

Conflict, redundancy and subsumption are defined above as logical conditions. These conditions can be detected if syntax allows one to examine two rules and determine whether situations exist in which both can succeed, and whether the results of applying the two rules are the same, conflicting, or unrelated.

4.3.2 Logical Checks for Completeness

Incanpleteness of the knowledge base is the result of:

MISSING RULES: a situation exists in which a particular result is required, but no rule can succeed in that situation to produce the desired result.

Missing rules can be detected logically if it is possible to enumerate all circumstances in which a given decision should be made or a given action should be taken.

4.3.3 Pragmatic Considerations

It is often pragmatic conditions, not purely logical ones, that determine whether there are true inconsistencies in a knowledge base. The semantics of the domain nay modify syntactic analysis. Of the three types of inconsistency described above, only conflict is guaranteed to be a true error.

In practice, logical redundancy may not cause problems. In a system where the first successful rule is the only one to succeed, a problem will arise only if one of two redundant rules is revised or deleted while the other is left unchanged. On the other hand, in a system using a scoring mechanism (such as certainty factors in EMYCIN systems), redundant rules cause the same information to be counted twice, leading to erroneous increases in the weight of their conclusion.

In a set of rules that accumulate evidence for a particular hypothesis, one rule which subsumes another may cause an error by counting the same evidence twice. Alternatively, the expert might have purposely written the rules so that the more restrictive one adds a little more weight to the conclusion made by the less restrictive one.

An exhaustive syntactic approach for identifying missing rules would assume that there should be a rule which applies in each situation defined by all possible combinations of a number of domain variables. Some of these canbinations, however, might not be meaningful. As with consistency, checking for completeness generally requires some knowledge of the problem domain.

Because of these pragmatic considerations, an automated rule-checker should display potential errors and allow an expert to indicate which ones represent real problems. It should prompt the expert for domain-specific information to explain why apparent errors are, in fact, acceptable. This information should be represented so that it can be used to make future checking more accurate.

5 Rule-Checking in ONCOCIN

5.1 Description of ONCOCIN

ONCOCIN is a rule-based consultation system to advise physicians at Stanford's Oncology Day Care Center on the management of patients who are on experimental treatment protocols. These protocols serve to ensure that data fran patients on various treatment regimens can be compared to evaluate the success of therapy and to assess the relative effectiveness of alternative regimens. A protocol specifies when the patient should visit the clinic, what chemotherapy and/or radiation therapy the patient should receive on each visit, when laboratory tests should be performed, and under what circumstances and in what ways the recanmended course of therapy should be modified.

A _rule_ in ONCOCIN is a production with an _action_ part that concludes a _value_ for some _parameter_ on the basis of values of other parameters in the rule's _condition_ part. Currently all parameter values can be determined with certainty; there is no need to use weighted belief measures. When a rule succeeds, its action parameter becomes _known_ so no other rules with the same action parameter will be tried.

Rules specify the context in which they apply. Examples of ONCOCIN contexts are drugs, chmotherapies (i.e., drug combinations), and protocols. A rule which determines the dose of a drug may be specific to the drug alone, or to both the drug and the chemotherapy. In the latter case, the context of the rule would be the list of pairs of drug and chemotherapy for which the rule is valid. At any time during a consultation, the current context represents the particular drug, chemotherapy, and protocol currently under consideration.

In order to determine the value of a parameter, the system tries rules which conclude about that parameter and which apply in the current context. For example, Rule 75 shown below is invoked to determine the value of the parameter "current attenuated dose" (point a), and when the current context is a drug in the chemotherapy MOPP, or a drug in the chemotherapy PAVe (point b).

```
                         RULE 75

[Action Parameter]    (a) To determine the current attenuated dose
         [Context]    (b) for all drugs in MOPP, or for all drugs in PAVe:

       [Condition]       If:  1) This is the start of the first cycle
                                 after a cycle was aborted, and
                              2) The blood counts do not warrant dose
                                 attenuation
          [Action]     Then:  Conclude that the current attenuated dose
                                 is 75 percent of the previous dose.
```

Certain rules for determining the value of a parameter serve special functions. Some give a "definitional" value in the specified context. These are called initial rules and are tried first. Other rules provide a (possibly context-dependent) "default" or "usual" value in the event that no other rule succeeded. These are called _default_ rules and are applied last. Rules which do not serve either of these special functions are called _normal_ rules. Concluding a parameter% value, then, consists of trying, in order, three groups of rules: initial, then normal, then default. A rule's _classification_ tells which of these three groups it belongs to.

Internally in LISP, the context, condition, action, and classification are properties of an atom representing the rule. The internal form of rule 75 is shown below.

```
RULE075
   CONTEXT:         ((MOPP DRUG)(PAVE DRUG))
   CONDITION:       (AND ($IS POST.ABORT 1)
                         ($IS NORMALCOUNTS YES))
   ACTION:          (CONCLUDEVALUE ATTENDOSE (PERCENTOF 75 (PREVIOUSDOSE))
   CLASSIFICATION: NORMAL
```

The LISP functions which are used in conditions or actions have _templates_ indicating what role their arguments play. For example, both $IS and CONCLUDEVALUE take a parameter as their first argunent and a value of that parameter as their second argument. Each function also has a _descriptor_ representing its meaning. For example, the descriptor of $IS shows that the function will succeed when the parameter value of its first argument is equal to its second argunent.

5.2 _Overview of the Rule-Checking Program_

A rule's context and condition together describe the situations in which it applies. The templates and descriptors of rule functions make it possible to determine the canbination of values of condition parameters which will cause a rule to succeed. The rule's context property shows the context(s) in which the rule applies. The context and condition of two rules can therefore be examined to determine if there are situations in which both can succeed. If so, and the rules conclude different values for the same parameter, they are in conflict. If they conclude the same value for the same parameter, they are redundant. If they are the same except that one contains extra condition clauses, then one subsunes the other.

These definitions of inconsistencies simplify the task of checking the knowledge base. The rules can be partitioned into disjoint sets, each of which concludes about the same parameter in the same context. The resulting rule sets can be checked independently. To check a set of rules, the program:

(1) finds all parameters used in the conditions of these rules;

(2) makes a table, displaying all possible canbinations of condition parameter values and the corresponding values which will be concluded for the action parameter'[1];

(3) checks the tables for conflict, redundancy, subsunption, and missing rules; then displays the table with a summary of any potential

[1]Because a parameter's value is always known with certainty and the possible values are mutually exclusive, the different canbinations of condition parameter values are disjoint. If a rule corresponding to one canbination succeeds, rules corresponding to other canbinations in the same table will fail. This would not be true in an EMYCIN consultation system in which the values of some parameters can be concluded with less than canplete certainty. In such cases, the canbinations in a given table would not necessarily be disjoint.

errors that were found. The rule checker assumes that there should be a rule for each possible combination of values of condition parameters; it hypothesizes missing rules based on this assumption[2].

ONCOCIN's rule-checker dynamically examines a rule set to determine which condition parameters are currently used to conclude a given action parameter. These parameters determine what columns should appear in the table for the rule set. The program does not expect that each of the parameters should be used in every rule in the set (as illustrated in by rule 76 in the example below). In contrast, TEIRESIAS examined the "nearly complete" MYCIN knowledge base and built static rule models showing (among other things) which condition parameters were used (in the existing knowledge base) to conclude a given action parameter. When a new rule was added to MYCIN, it was compared with the rule model for its action parameter. TEIRESIAS proposed missing clauses if some condition parameters in the model did not appear in the new rule.

5.3 An Example

ONCOCIN's rule checking program can check the entire rule base, or can interface with the system's knowledge acquisition program and check only those rules affected by recent changes to the knowledge base. This latter mode is illustrated by the example in Fig. 1; the system builder is trying to determine whether the recent addition of one rule and deletion of another have introduced errors.

[2] We plan to add a mechanism to acquire information about the meaning of parameters and the relationships among them, and to use this information to omit semantically impossible can binations from subsequent tables.

The rules checked in the example conclude the current attenuated dose for the drug cytoxan in the chemotherapy CVP. There are three condition parameters commonly used in those rules. Of these, NORMALCOUNTS takes "YES" or "NO" as its value. CYCLE and SIGXRT take integer values. The only value of CYCLE or SIGXRT which was mentioned explicitly in any rule is "1"; therefore, the table has rows for values "1" and "OTHER" (i.e., other than 1).

The table shows that rule 80 concludes that "attenuated dose" should have the value "250 milligrams per square meter" when the blood counts do not warrant dose attenuation (NORMALCOUNTS = YES), the chemotherapy -cycle nunber is 1 (CYCLE = 1), and this is the first cycle after significant radiation (SIGXRT = 1). This canbination of values of the condition parameters is labeled Cl.

Rule 76 can succeed in the same situation (Cl) as rule 80, but it concludes a different dose. These rules do not conflict, however, because rule 76 is a "default" rule which will be invoked only if all "normal" rules (including rule 80) fail. Note that NORMALCOUNTS is the only condition parameter which appears explicitly in rule 76, as indicated by the parentheses around values of the other two parameters. Rule 76 will succeed in all combination which include NORMALCOUNTS = YES (namely Cl, C3, C5, and C7).

Rules 667 and 67 are redundant because both use combination C2 to conclude the value labled V2 (250 mg/m^2 attenuated by the minimum count attenuation).

Rule 600 is in conflict with rule 69 because both use canbination C6, but they conclude different values (and both are categorized as "normal" rules).

No rules exist for combinations C4 and C8, so the program hypothesizes that rules are missing.

Rule set: 667 600 82 80 69 67 76

Context: the drug CYTOXAN in the chemotherapy CVP

Action Parameter: the current attenuated dose

Condition Parameters:
NORMALCOUNTS - the blood counts do not warrant dose attenuation
CYCLE - the current chemotherapy cycle number
SIGXRT - the number of cycles since significant radiation

Values too long to appear in the Value column:
V1 - the previous dose advanced by 50 mg/m^2
v2 - 250 mg/m^2 attenuated by the minimum count attenuation
V3 - the minimum of 250 mg/m^2 and the previous dose
V4 - the minimum of 250 mg/m^2 and the previous dose attenuated by the minimum count attenuation

Evaluation	Rule	Value	NORMALCOUNTS	CYCLE	SIGXRT	Combination
	80	250 mg/m^2	YES	1	1	C1
	76 (D)	Vi	YES	(1)	(1)	C1
R	667	v2	NO	1	1	c2
R	67	v2	NO	1	1	c2
	76 (D)	V1	YES	(1)	(OTHER)	C3
M	.		NO		OTHER	C4
	82	V3	YES	OTHER	1	C5
	76 (D)	V1	YES	(OTHER)	(1)	C5
C	600	V3	NO	OTHER	1	C6
C	69	v4	NO	OTHER	1	C6
	76 (D)	V1	YES	(OTHER)	(OTHER)	c7
M	.		NO	OTHER	OTHER	C8

SUMMARY OF COMPARISON

Conflict exists in combination(s):	C6 (RULE600 RULE069)
Redundancy exists in combination(s):	C2 (RULE667 RULE067)
Missing rules are in combination(s):	C4, C8

NOTES

Evaluation:
M - Missing; C - Conflict; R - Redundant.

Rules:
Default rule are indicated by (D).

Values of Condition Parameters:
A value in parentheses indicates that the parameter is not explicitly used in the rule, but the rule will succeed when parameter has this value.

Figure 1. An Example of the Rule-Checking Program

The system builder can enter ONCOCIN's knowledge acquisition program to correct any of the errors found by the rule-checker. A missing rule can be displayed in either LISP or English (Fig. 2), and added to the system's knowledge base after the expert has provided a value for its action parameter.

<u>Missing rule corresponding</u> to <u>combination C4:</u>

```
To determine the current attenuated dose for Cytoxan in CVP:
   If:  1) The blood counts do warrant dose attenuation,
        2) The current chemotherapy cycle nunber is 1, and
        3) This is not the start of the first cycle after
           significant radiation
 Then : Conclude that the current attenuated dose is .......
```

<u>Figure 2</u>. Proposed Missing Rule (English Translation)

Note that no value is given for the action parameter; this could be filled in by the system builder if the rule looked appropriate for addition to the knowledge base.

If a summary table is too big to display, it is divided into a nunber of subtables by assigning constant values to some of the condition parameters. If the conditions involve ranges of numeric values, the table will displays these ranges graphically as illustrated in Fig. 3.

```
            Rule set:  33 24

             Context:  the drug DTIC in the chemotherapy ABVD

    Action Parameter:  the dose attenuation due to low WBC

       Default value:  100

Evaluation Rule    Value       WBC (in thousands)    Canbination
                (percentage)
                               0  1.5  2  3  5
           33        25        ....****0......           C1
           24        50        ........***0..            c2

SUMMARY OF COMPARISON
   No problems were found.

NOTES
   *'s appear beneath values included by the rule
   0's appear beneath upper or lower bounds that
        are not included.
   E.g., Rule 33 applies when  1.5 <= WBC < 2.0
```

Figure A Table of Rules with Ranges of Numerical Values

5.4 Effects of the Program

The rule checking program described in this paper was developed at the same time that ONCOCIN's knowledge base was being built. During this time, periodic runs of the rule checker suggested missing rules that had been overlooked by the oncology expert. It also detected conflicting and redundant rules; these generally arose because a rule had the incorrect context and therefore appeared in the wrong table.

A nunber of inconsistencies in the use of domain concepts were revealed by the rule checker. For example, on one occasion the program proposed a missing rule for a meaningless combination of condition parameter values. In discussing the domain knowledge that expressed the interrelationship among the values, it became clear that a nunber of individual yes/ no valued parameters really could be represented more logically as different values for the same parameter.

The knowledge engineers and oncology experts alike have found the rule checker's tabular display of rule sets much easier to interpret than a rule-by-rule display. Having tabular summaries of related rules has facilitated the task of modifying the knowledge base.

6 Concluding Remarks

The program we have described assists a knowledge engineer in ensuring the consistency and canpleteness of the rule set in the ONCOCIN rule-based consultation system. The program has already proved useful in development of that system. The program's design is general so that it could be adapted to other rule-based systems.

References

1. Davis, R. Applications of Meta-level Knowledge to the Construction, Maintenance, and Use of Large Knowledge Bases. Doctoral dissertation, Computer Science Department, Stanford University, June 1976.

2. Shortliffe, E.H. Cunputer-Based Medical Consultations: MYCIN. Elsevier/North Holland Publishing Canpany, New York, 1976.

3. Shortliffe, E.H., Scott, A.C., Bischoff, M.B., Campbell, A.B., van Melle, W., and Jacobs, C.D. ONCOCIN: An expert system for oncology protocol management. Proceedings of 7th IJCAI, pp. 876-881, Vancouver, B.C., August 1981.

4. van Melle, W. A Domain-Independent System that Aids in Constructing Knowledge-Based Consultation Programs. Doctoral dissertation, Computer Science Department, Stanford University, June 1980.