Report 81-05
Stanford -- KSL

Scientific DataLink

An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System. Motoi Suwa, A. Carlisle Scott, Edward H. Shortliffe, Jun 1981

Stanford Heuristic Programming Project **June 1981**
HPP-81-5

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, CA 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) and the National Science Foundation (Grant MCS 7903753). It was carried out on the SUMEX-AIM computer (NIH grant RR 00785).

* Present Address: Computer Vision Section
Electrotechnical Laboratory
1-1-4 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 most 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 ONCOCIN research project. We would like to thank all of those who helped to make the program possible, and to thank Craig Tovey for valuable comments on earlier drafts of this paper.

BLANK PAGE

1 Introduction

When constructing a knowledge-based program, a system-builder must ensure that a user who requests assistance will be given accurate advice or the correct solution. The process of verifying that a system is accurate and reliable has two distinct components: 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.

Validation of a knowledge-base includes testing and refining the system's knowledge until it is judged correct and complete, a process which is one component of the larger problem of knowledge acquisition. Knowledge-base refinement (debugging) involves discovering and correcting a variety of errors (bugs) 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 ONCOCIN [3], a recently-developed rule-based expert system.

2 Knowledge Acquisition

Before knowledge can be embodied in a computer system, it must

undergo a number of transformations. First, a human acquires expertise in some domain through study, research and experience. The expert then attempts to formalize this expertise and express it using the internal representation of a consultant program, viz. production rules, frames, or semantic nets. Finally, the knowledge, represented as LISP expressions or in some other machine-readable form, 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 computer are frequently the 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 could be facilitated by a system which:

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

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

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

An automated assistant designed to aid 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, however, and no formal assessment of rules occurred at the time they were initially entered into the knowledge base.

In EMYCIN [4], our system for building knowledge-based consultants, 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 review and correction of the system's conclusions (a generalization of the TEIRESIAS program),

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

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

4.2 Systematic Checking of a Knowledge Base

The knowledge-base debugging tools mentioned in the previous section concentrated on running test cases and allowing a system builder to identify problems rather than on systematically checking the knowledge base. One problem with this approach is that it is rarely possible to guarantee that a knowledge-base is completely debugged, even after hundreds of test runs. While running test cases is an essential part of verifying the consistency and completeness of a knowledge base, the debugging process can be usefully

augmented by systematically checking for common errors that arise during knowledge acquisition.

Only in special cases is it possible to test a growing knowledge base by running sample cases. For example, TEIRESIAS was developed after the MYCIN system and an extensive rule set had already been built. EMYCIN is specifically designed for 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 requires the selection of methods to represent the domain knowledge and the design of a program to use that knowledge before 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.

4.3 Debugging a Rule-Based System

When knowledge is represented in production rules, a number of possible errors in the knowledge base can generally be classified into one of four categories: conflict, redundancy, subsumption, or missing rules. These issues may be summarized as follows:

CONFLICT: two rules succeed under the same circumstances but make contradictory conclusions.

REDUNDANCY: two rules make the same conclusion in the same situation. This may cause problems in performance, depending on the particular rule interpreter used. In a system where the first correct rule is the only one to succeed, no problem arises unless one of the two rules is revised or deleted and the other left unchanged. On the other hand, if the system uses a scoring mechanism (such as certainty factors used in EMYCIN systems), the same information would be counted twice leading to erroneous increases in the weight of the conclusion.

SUBSUMPTION: two rules are similar except that one contains additional restrictions on the situations in which it will succeed. Whenever the more restrictive rule succeeds, the less restrictive rule also will succeed, resulting in redundancy.

MISSING RULES: there is a situation in which a particular decision will need to be made, but no rule exists which applies in that situation to make an appropriate conclusion.

Some or all of these problems can be detected, depending on the nature of the rules used in the system. Conflict, redundancy and subsumption can be detected if the form of the rules is simple enough; one can examine the structure of two rules and determine whether there exist situations in which both can succeed and whether the results of applying the two rules are similar, contradictory, or unrelated. Missing rules can be detected if it is

possible to enumerate all circumstances in which a given decision should be made or a given action should be taken.

When rule syntax permits detection of these kinds of errors in a rule set, a systematic verification procedure can be implemented. System builders rarely have the time or patience to examine all rules, and are likely to overlook some errors that could be detected effectively by a systematic search.

The verification process may not be purely syntactic, however. In most cases automated checking will need some knowledge of the problem domain. For example, when looking for missing rules, a system may try to find rules that will apply in a set of situations defined by all possible combinations of a number of factors, not realizing that some of these combinations are not meaningful.

In many cases only the expert can say whether an error really exists. For example, consider a set of rules which accumulate evidence for a particular hypothesis and in which one rule subsumes another. In some cases, this will cause an error because the same evidence will be counted twice. In other cases, the expert may have purposely structured the knowledge so that the more restrictive rule adds a little more weight to the conclusion made by the less restrictive rule.

For these reasons, a program to check the rule set correctness should have mechanisms for displaying potential errors, for allowing an expert to indicate which represent real problems, and for prompting the expert for

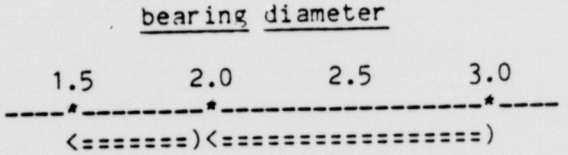
domain-specific information when told that an apparent error is actually acceptable. In future runs of the program, the system could use the domain-specific knowledge to make more accurate assessments when considering a potential error in the rule set.

It is also important that a computer-based assistant be able to communicate in terms that are easy for the expert to understand. Despite the apparent simplicity and clarity of production rules, especially when they are translated into English, the human expert may have difficulty understanding the meaning of a rule or problems of interactions among rules. For example, in discussing two rules which apply in the situations:

(1) If: the diameter of the bearing is greater than or equal to 2.0 cm and less than 3.0 cm

(2) If: the diameter of the bearing is greater than or equal to 1.5 cm and less than 2.0 cm,

it might be more useful to illustrate the situations graphically:



5 Rule-Checking in ONCOCIN

5.1 Description of ONCOCIN

ONCOCIN is a rule-based consultation system under development at Stanford University [3]. Its purpose is 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 from patients on various treatment regimens can be compared to evaluate the success of therapy and to assess the relative effectiveness of alternatives. 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 recommended course of therapy should be modified.

Rules in ONCOCIN are productions similar in format to those used in EMYCIN systems. A rule's "action" part concludes a value for some attribute or parameter on the basis of values of other parameters in the rule's "condition" part. Currently ONCOCIN only uses parameters whose values can always be determined with certainty. Consequently, each clause in a rule's condition will be true, false, or unknown when the rule is tried; there is no need to use weighted belief measures.

Rules are categorized by the "context" in which they apply. Examples of ONCOCIN contexts are drugs and the chemotherapies (i.e., drug

combinations) in which the drugs are given. A rule which determines the dose of a drug may be specific to the drug alone, or to both the drug and the chemotherapy. The context of the rule, then, is the list of pairs of drug and chemotherapy for which the rule is valid.[1] At any time during a consultation, the "current context" is the system's representation of the particular drug and chemotherapy currently under consideration.

When the system needs to determine the value of a parameter, it tries each rule which concludes about that parameter and which applies in the current context. For example, RULE075 shown below is invoked (a) to determine the value of the parameter "current attenuated dose", and (b) when the current context is a drug in the chemotherapy MOPP, or a drug in the chemotherapy PAVe.

RULE075

[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.

There are three steps in determining the value of a parameter. First the system checks to see if the value can be determined by definition in the current context. If not, the "normal" method of finding the value is used: if the parameter corresponds to a piece of laboratory or other data that the

[1] In MYCIN and other EMYCIN-based systems, such contextual information was specified by extra clauses at the beginning of a rule's condition part.

user is likely to know, it is requested of the user; otherwise, rules for concluding the parameter are tried. Finally, the system may be provided with a (possibly context-dependent) default value which will be used in the event that the normal mechanism fails to produce a value, or a question may be asked so that the user can provide the default value. Each rule contains a classification ("initial", "normal", or "default") indicating the steps in which the rule will be used.

5.2 Overview of the Rule-Checking System

Checking ONCOCIN's rules for consistency and completeness consists of three steps:

(1) partition all rules into sets of rules which make conclusions about the same parameter in the same context,

(2) make a table for each rule set, displaying the combinations of parameter values from the conditions of the rules which will lead to each of the possible values for the parameter about which the rules conclude[2], and

(3) check the table for conflict, redundancy, and missing rules[3].

[2]Because a parameter's value is always known with certainty, the different combinations of condition parameter values are disjoint. If a rule corresponding to one combination succeeds, rules corresponding to other combinations in the same table will not.

[3]The checking process would be more complicated in an EMYCIN consultation system in which the values of some parameters can be concluded with less than complete certainty. In such cases, the combinations in a given table would not necessarily be disjoint.

As discussed earlier, problems of conflict and redundancy can be detected if it is possible to tell whether rules can succeed in the same situation and whether they make conclusions which are similar or contradictory. ONCOCIN's rules meet these two requirements. A rule's context and condition together describe the situation in which it applies. The conclusions of two rules are similar if they conclude the same value for the same parameter; they are contradictory if they conclude different values for the same parameter.

Checking ONCOCIN's rules for consistency, therefore, can be accomplished by analyzing in turn each of the disjoint sets of rules that has been partitioned using the following five sub-steps:

(1) Look for a parameter whose value is concluded by rules.

(2) Get the list of rules which conclude about that parameter.

(3) Divide the rules by contexts.

(4) Divide the rules into groups which share at least one common parameter in the condition portion.

(5) Merge small groups that contain rules in common.

The partitioning process results in a number of rule sets, each containing rules which test the values of the same "condition" parameters in order to determine the value of the same "action". The second step of rule checking is to build a table which summarizes the values of the "condition" parameters as they relate to the the value of the "action" parameter in the

various rules of the partitioned subset. The rule checker assumes that there should be a rule for each possible combination of values, and it uses this assumption to suggest instances where rules may be missing from the knowledge base.

Our system currently generates all possible combinations of condition parameter values because it contains no knowledge about the meaning of these parameters or the relationships among them. However, when the knowledge base contains a rule stating that one value for a certain parameter implies a specific value for another parameter, it subsequently will suppress the generation of nonexisting combinations when building its tabular summaries.

The next step for the program is to check the table for contradictory, redundant, or missing rules. When this has been done, it displays the table, including a summary of any potential errors that were found.

If there is a rule missing from the table, the system builder can ask for a new rule to be generated. A rule will be written which is complete except for the value of the action parameter. This can be added to the rule using ONCOCIN's knowledge acquisition program (adapted from EMYCIN's knowledge-base editor [4]).

5.3 An Example

The system can check the entire rule set, or can interface with ONCOCIN'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 the deletion of another has caused problems with ONCOCIN's rule set.

The rules that were checked in this example are invoked in order to determine 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 one of two values, "YES" or "NO". CYCLE and SIGXRT take integer values (in this particular example, the only value mentioned explicitly in any rule was 1; therefore, the table generator made rows for "1" and "other than 1").

The table shows that RULE080 concludes the value "250 mg/m2" for attenuated dose when the values of the condition parameters are NORMALCOUNTS = YES, CYCLE = 1, and SIGXRT = 1.

Although RULE080 and RULE076 use the same combination of values to conclude different doses, they do not conflict because RULE076 is a "default" rule which will be invoked only if all "normal" rules (including RULE080) fail.

The summary of the results is shown at the bottom of the figure.

Rule set: 667 600 82 80 69 67 76

Parameter concluded in these rules: the current attenuated dose
Context: the drug CYTOXAN in the chemotherapy CVP

Ev	Rule	Value	NORMALCOUNTS	CYCLE	SIGXRT	Combination
	80	250mg/m2	YES	1	1	1
	76 (default)	1	YES	(1)	(1)	1
&	667	2	NO	1	1	2
&	67	2	NO	1	1	2
	76 (default)	1	YES	(1)	(OTHER)	3
+	.		no	1	other	4
	82	3	YES	OTHER	1	5
	76 (default)	1	YES	(OTHER)	(1)	5
>	600	3	NO	OTHER	1	6
>	69	4	NO	OTHER	1	6
	76 (default)	1	YES	(OTHER)	(OTHER)	7
+	.		no	other	other	8

NOTES

Ev (Evaluation): + - Missing Rule > - Conflict & - Redundant

Values too long to appear in the Value column:
1 - the previous dose advanced by 50 mg/m2
2 - 250 mg/m2 attenuated by the minimum count attenuation
3 - the minimum of 250 mg/m2 and the previous dose
4 - the minimum of 250 mg/m2 and the previous dose attenuated by the minimum count attenuation

Condition Parameters:
NORMALCOUNTS - The blood counts do warrant dose attenuation
CYCLE - The current chemotherapy cycle number
SIGXRT - The number of cycles since significant radiation

Notation for Values of Condition Parameters:
YES - the parameter is actually used in the rule.
(YES) - the rules succeeds when parameter has this value although the parameter is not explicitly used in the rule.
yes - there is no rule corresponding to this combination of values.

Combination - the number corresponding to the combination of values for the condition parameters.

SUMMARY OF COMPARISON
Conflict exists in combination(s): 6 (RULE600 RULE069)
Redundancy exists in combination(s): 2 (RULE667 RULE067)
Missing rules are in combination(s): 4, 8

Figure 1. An Example of the Rule-Checking System

The system builder can request the display of a missing rule, in either LISP or English (Fig. 2), and enter ONCOCIN's knowledge acquisition program to add this new rule to the system's knowledge base after providing a value for its action parameter (the "current attenuated dose" in Fig. 2).

Missing rule corresponding to combination 4:

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

Figure 2. 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 otherwise looked appropriate for addition to the knowledge base.

If a summary table is too big to display, it is divided into a number 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

Parameter concluded in these rules: the dose attenuation due to low WBC
Context: the drug DTIC in the chemotherapy ABVD
Default value: 100

Ev	Rule	Value (percentage)	WBC (in thousands) 0 1.5 2 3 5	Combination
	33	25	....****0.....	1
	24	50	.....***0...	2

Notation: *'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 3. A Table of Rules with Ranges of Numerical Values

5.4 Effects of the System

The Rule Checking system described in this paper was developed at the same time that ONCOCIN's knowledge base was being built. During the time when both rule checker and knowledge base were developing, periodic runs of the rule checker on the knowledge base were very useful in suggesting missing rules that had been overlooked by the Oncology expert, and in detecting conflicting and redundant rules. Generally, these latter problems existed because a rule had been given the wrong context property and therefore appeared in the wrong table.

A number of inconsistencies in the use of domain concepts were

revealed by the rule checker and subsequently fixed. For example, on one occasion the program proposed a missing rule for a combination of condition parameter values that was not possible. In discussing the domain knowledge that expressed the interrelationship among the values, it became clear that a number of individual yes/no parameters really should have been represented 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 rule-by-rule display. This mechanism for explicitly displaying related rules in a summary form 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 completeness 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, however, and it could be adapted to other rule-based systems. The simple syntax of the ONCOCIN rule language has facilitated the development of the program, but substantial modifications would be needed if the rule-checker were to be used in a domain where rules included certainty weights to qualify the strength of inference.

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. Computer-Based Medical Consultations: MYCIN. Elsevier/North Holland Publishing Company, New York, 1976.

3. Shortliffe, E.H., Scott, A.C., Bischoff, M., Campbell, A.B., van Melle, W., Jacobs, C. ONCOCIN: An Expert System for Oncology Protocol Management. Submitted for publication in the Proceedings of IJCAI-81, 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.