Report 84-38
Stanford -- KSL

Scientific DataLink

Enhancing Performance of Expert Systems
by Automated Discovery of Meta-Rules.
Li-Min Fu, Bruce G. Buchanan,
Sep 1984

card 1 of 1

Heuristic Programming Project
Report No. HPP 84-38

September 1984

Enhancing Performance of Expert Systems by Automated Discovery of Meta-Rules

Li-Min Fu
and
Bruce G. Buchanan

6 September 1984

Computer Science Department
Stanford University
(415) 497-0935

Table of Contents

ENHANCING PERFORMANCE OF EXPERT SYSTEMS BY AUTOMATED DISCOVERY OF META-RULES

Abstract

Machine learning can be used to formulate new meta-level knowledge. A small MYCIN-like medical diagnosis system was constructed as a starting point. Two heuristic methods are used in a program called Meta-Rulegen to form meta-rules from the knowledge base in the diagnosis system. In a preliminary study, 63 meta-rules were formed automatically and, by judiciously selecting a set of meta-rules, the efficiency of the diagnosis system can be improved significantly without degrading the quality of advice. This study suggests that meta-rules can be learned automatically to improve the efficiency of rule-based systems.

1 Introduction

The value of meta-level knowledge for guiding the invocation, construction, and explanation of object-level rules in an expert system has been demonstrated by Davis [2]. In this paper we explore the use of machine learning methods for formulating new meta-level knowledge, extending Davis' ideas about learning rule models. We have constructed a small MYCIN-like medical diagnosis system as a starting point. One line of research (not reported here) is to learn new diagnostic rules from examples. The research reported here aims at learning meta-rules that will guide the rule-based diagnostic system by pruning and reordering the diagnostic rules, as in MYCIN [3].

We are strongly motivated by the fact that meta-rules are important in systems with large knowledge bases to avoid exhaustive search. Yet, human experts should not be concerned with control issues as much as with domain knowledge, so it is desirable to automate the formulation of control rules.

The performance program, called JAUNDICE, is a rule-based consultant with 141 diagnostic rules that diagnoses likely causes of jaundice,[1] and 80 non-diagnostic rules that are linked in a network reflecting a taxonomy among diseases and causal links among events as suggested in [5], [8] and [1].

---

[1] The performance program is for preliminary screening of jaundice cases, without recourse to results of invasive tests. Its performance has been tested preliminarily by 72 Jaundice cases collected from the literature and textbooks, and the diagnostic accuracy is 84.7%.

Each rule has the same form as a rule in EMYCIN [7].[2]

Note that this paper reports on a 2nd-order learning problem (and not on the first-order problem of learning new object-level rules) which is defined as:

Given:
A set of object-level rules including:
inferential, causal, and taxonomic knowledge of
the domain.

Find:
Useful meta-rules that improve system efficiency
by guiding the invocation of object-level rules.

This paper first discusses design considerations and then presents implementation details of the 2nd-order learning system with preliminary results.

2 Learning Meta-Rules: Design Considerations

2.1 Format of Meta-Rules

As in [2], we use two syntactic forms of meta-rules: pruning and reordering forms (see Figures 1 and 2). In fact, the distinction between these two forms is often blurred semantically. For instance, if we say "Do Rule-Set1 before Rule-Set2", and we succeed in our goal by invoking only Rule-Set1, then Rule-Set2 is pruned anyway. As seen in Figure 1 or 2, the premise of a meta-rule has three parts. The first part is the goal description which can be global, local, or curren, and, in our system, the global goal is disease-entity, and local goals include: syndrome, pathophysiological mechanism, etiology, etc. ; the second part is a conjunction in which each conjunct is a predicate with a triplet of attribute, object, value; and the third part is the description of concluded rule sets. The description of the rule sets in the third parts of meta-rules may be by content or by name. Indirect referencing (by content) is important to maintain flexibility and understandability. The learning program also expands indirect references into a so-called name-referred form with an explicit list of rule names, as seen in figure 7.

---

[2] But, we did not start with EMYCIN because we wanted to keep the performance program simple and sparse. More importantly, we wanted to augment the syntax of the knowledge, as described above.

```
Meta-rule001
If  1. The goal is to conclude the disease of
       Jaundice.
    2. The indirect type bilirubin is not
       dominant.
    3. There are rules which mention in their
       premise "overproduction of bilirubin".
Then it is definite(1.) that each of these rules
     is not going to be useful.

Premise:
($AND (SAME JAUNDICE GOAL DISEASE-ENTITY)
      (DIFFER LFT DOMINANT-BILIRUBIN INDIRECT)
      (THEREARE OBRULES ($AND (MENTIONS PREMISE
         OVERPRODUCTION-OF-BILIRUBIN)) SET1))
Action:
(CONCLUDE SET1 UTILITY NO 1.)
```

Figure 1. Example of a meta-rule in pruning form created by META-RULEGEN. Upper part is it's English translation; lower half is it's code in INTERLISP.

```
Meta-rule079
If  1. The goal is to conclude the disease
       mechanism of Jaundice.
    2. The Alkaline Phosphatase level in serum
       is greater than 15 B.U.
    3. There are rules which mention in their
       action "Cholestasis".
    4. There are rules which mention in their
       action "Parenchymal-dysfunction".
Then it is probable(.8) that the former should
     be invoked before the latter.

Premise:
($AND (SAME JAUNDICE GOAL DISEASE-MECHANISM)
      (SAME LFT ALKALINE-PHOSPHATASE >15B.U.)
      (THEREARE OBRULES ($AND (MENTIONS ACTION
         CHOLESTASIS)) SET1)
      (THEREARE OBRULES ($AND (MENTIONS ACTION
         PARENCHYMAL-DYSFUNCTION)) SET2))
Action:
(CONCLUDE SET1 DOBEFORE SET2 .8)
```

Figure 2. Example of a meta-rule in reordering form created by META-RULEGEN.

2.2 Utility Consideration of Meta-Rules

The main reason for incorporating meta-rules in a performance system is efficiency.[3] Theoretically, it is difficult to say how to measure the efficacy of meta-rules unless we delineate the concept of Utility Value for the meta-rules, which is also important in generating them.

2.2.1 Utility Value for Meta-Rules

The Utility Value of meta-rules is based on an analysis of costs and benefits. Intuitively, high Utility Value is associated with high benefit and low cost. We define an absolute utility value based on estimated savings in CPU time and then a relative value to normalize absolute values over object rule sets of different sizes.

Absolute Utility Value of Meta-Rules

The cost of a meta-rule is the estimated CPU time to evaluate its premise. If the premise is true, then its benefit is how much CPU time might be saved by pruning or reordering object level rules under the guidance of the meta-rule. For simplicity, we first define unit cost to be "average CPU time to evaluate one conjunct (clause) in the premise." Suppose there are n conjuncts in the premise of the meta-rule, we define the cost of the meta-rule to be n units. (The performance program may cease evaluation once one conjunct appears to be false.but our estimate will assume the worst case.) The benefit will be how many object level rules are pruned out, if the meta-rule succeeds (i.e., if it's premise is true). The cost of using one object level rule will include CPU time for evaluating its premise, and, if the premise is true, making a conclusion and doing the bookkeeping. But, considering the least condition (i.e., first conjunct is found to be false, and evaluation is stopped), if there are m object level rules which are pruned away by the meta-rule, then the least benefit will be m units. Again, our estimate assumes the worst case (i.e.,least benefit). Thus, if a meta-rule makes a successful and accurate prediction, then the gain (the ratio of benefit to cost) will be "m/n". Now, the Absolute Utility Value (abbreviated as AUV) is defined for a meta-rule as follows:

---

[3] We are not concerned here with the use of meta-rules to guide a dialogue, although human engineering issues are also important. If the meta-rules are effective in pruning unnecessary questions, however, the dialogue will also appear to be better focused.

$$AUV = b/c_o \times Freq.(prem.) \times C_{mR}.$$

where,
- b = Number of object rules pruned.
- c_o = Number of conjuncts in the premise.
- $Freq.(prem.)$ = the estimated frequency with which the premise is true.
- C_{mR}[4] = degree of certainty of meta-rule.

This definition of AUV takes into account not only the cost and benefit, but also the frequency with which the premise is true over reference cases in a case library (see Figure 6) and the degree of certainty of the meta-rule. If the meta-rule rarely succeeds, it will have low utility. And, even if it succeeds (i.e., the premise is true), the prediction (conclusion) will be very uncertain if the degree of certainty of the conclusion is very low. When the frequency information is incomplete, it can be estimated in a Frequency Table that stores the number of cases (in the case library) for which each single premise clause is true. For instance, if a premise mentions the presence of Attribute1 and Attribute 2, then we can estimate the frequency of the conjunction by multiplying the frequency of Attribute1 and Attribute2 in the reference cases under the assumption (default) of independence. An example of calculating AUV is shown in section 4.

Theorem 1: The threshold value for AUV such that the expected benefit will be greater than zero is:

$$AUV_{threshold} = 1 + c_p f(1 - C_{mR})/c_o$$

- c_p: Penalty owing to the incorrect prediction by the meta-rule.[5]
- c_o: Number of conjuncts in the premise. This is the required cost to evaluate a meta-rule.
- b: Number of object rules pruned. This is the benefit when a meta-rule succeeds.
- f: $Freq.(prem.)$
- b_{exp}: Expected benefit of the meta-rule.

---

[4] C_{mR} ranges from 0 to 1. "Zero" means "unknown", while "one" means "definitely yes". It is obtained from human experts, or can be computed as described in section 3.2.

[5] If C_{mR} is not 1, then the meta-rule may predict wrong sometimes. And, the wrong predictions may cause penalty which depends on the extent the system undoes and re-executes, or the extent of improper reordering.

(proof): C_{mR} can be viewed as the estimated rate of correct predictions out of all predictions. Therefore, if the meta-rule succeeds (i.e., the premise is true) and the prediction is correct, then the net benefit will be "$(b - c_o)$", and if the meta-rule succeeds and the prediction is wrong, then the net benefit will be "$(-c_o - c_p)$". Otherwise, the net benefit is "$-c_o$" Hence, the expected benefit is:

$$b_{exp} = [(b-c_o)C_{mR} - (c_o+c_p)(1-C_{mR})]f - c_o(1-f)$$

Let $b_{exp} = 0$, we get $AUV = 1 + c_p f(1 - C_{mR})/c_o$, and this is the threshold value $AUV_{threshold}$.

Lemma 1: If $C_{mR} = 1$, then $AUV_{threshold} = 1$.

Lemma 2: If $c_p = 0$, then $AUV_{threshold} = 1$.

Both Lemma 1 and 2 are very useful, because the threshold value is a constant 1.

Lemma 3: If $AUV >> AUV_{threshold}$, then AUV can estimate the lower bound of expected benefit.

(proof): from proof of Thm. 1,

$$b_{exp}/c_o = AUV - AUV_{threshold}$$

If $AUV >> AUV_{threshold}$,

then $b_{exp}/c_o \simeq AUV$

since $c_o \geq 1$, AUV can estimate the lower bound

of expected benefit without knowing c_o.

Relative Utility Value of Meta-Rules

We also define Relative Utility Value (abbreviated as RUV) as follows.

$$RUV = \frac{AUV}{\text{Number of total object rules under the global goal}} \times 100$$

Three important aspects of RUV are:

1. RUV can estimate the relative improvement (in percentage) of the overall system performance. Because the overall cost for system execution is parallel to the number of total object rules, and , from Lemma 3 , AUV can estimate the expected benefit of a meta-rule , the ratio of the two can estimate the relative improvement of performance by a meta-rule.

2. RUV is good for comparison of meta-rules from different systems, which have different numbers of object rules.

3. RUV is more sensitive to reflect the real value of meta-rules. Because the object rule set usually expands quickly as a performance program is being constructed, RUV is a more important index to maintain useful meta-rules.

2.2.2 Selecting Useful Meta-Rules

Our heuristics for selecting useful meta-rules are as follows.

1. First, we use $AUV_{threshold} = 1$.

2. Second, we use RUV to do further pruning.

Then, we select a useful set of meta-rules by removing redundant[6] meta-rules and from experimental simulations.

2.3 Overview of Two Approaches to Learning Meta-Rules

2.3.1 From object rules

Starting from each object rule, the program uses information about three well-known medical strategies [4] to determine if there could be useful meta-rules covering this object rule and related ones.

1) Rule-out mechanism: If there exists enough evidence contradicting a fact, then we don't bother trying to confirm it or deduce other facts from it. If some evidence is against a fact, then we attempt to form a hypothesis: "That fact and all possibly associated facts with it may be ruled out based on that evidence", and if this hypothesis is justified by some evaluation criteria (e.g., Utility Value), then we succeed in our attempt.

---

[6] If two meta-rules often succeed simultaneously and their conclusions severely overlap, then they are redundant with each other. Both the frequency threshold and the extent of overlapping are defined heuristically.

2) Rule-in mechanism: This is the inverse of the Rule-out mechanism. It says we should consider certain facts first if some evidence implies doing so and this consideration is valuable with respect to some evaluation criteria.

3) Differential mechanism: Physicians are often involved with the issue of differential diagnosis. which is basically confirming one diagnosis from a set of possibilities to the exclusion of the others. In part. this is the consideration that when there is more evidence suggesting Disease1 than Disease2. it will be reasonable to confirm Disease1 before Disease2.

2.3.2 From attributes

If one conjunct appears many times in the premise of object level rules. then it might be worthwhile to evaluate this conjunct first. If this consideration proves valuable. then we keep it. A similar syntactic approach can be found in [2] and [7]: however. the difference is our explicit consideration of utility. as described in section 3.

3 Implementation

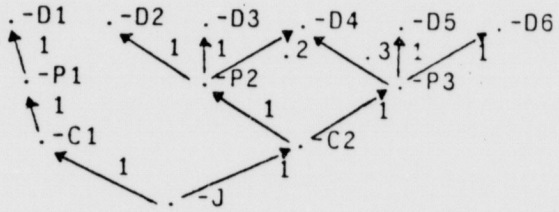
3.1 Overview of Meta-Rulegen

Meta-Rulegen is a second order learning program. which is written in Interlisp and runs on the SUMEX-AIM. Tops-20 system. The learning of meta-rules is based on:

1. 141 object level diagnostic rules. which are the inferential knowledge base of JAUNDICE. a Mycin-like consultation system that aids in the diagnosis of causes of jaundice.

2. Pathophysiological Taxonomy and causal links coded into 80 non-diagnostic rules: from these three semantic structures are built up:

    a. Denying-tree (see Figure 3): a network of disconfirming associations. If one node is denied. then. propagating along this tree. we know the degree of denial of relevant nodes. The denying tree is constructed by recording pairs of individual facts that are negatively linked in the object-level rules. For example. no overproduction of bilirubin (-P1) disconfirms (1.0) hemolysis (D1).



J: Jaundice
C1: Indirect bilirubin dominant
C2: Direct bilirubin dominant
P1: Overproduction of bilirubin
P2: Hepatocellular dysfunction
P3: Cholestasis
D1: Hemolysis
D2: Acute hepatitis
D3: Chronic hepatitis
D4: Neoplasm
D5: Calculous jaundice
D6: Primary biliary cirrhosis

Figure 3. Part of Denying-tree in META-RULEGEN.

b. Affirming-tree (see Figure 4): a network of confirming associations. If one node is suggested, then, by following the links in the affirming tree we know the degree of implication of other nodes. The affirming tree is constructed by recording pairs of individual facts that are positively linked in the object-level rules. For example, overproduction of bilirubin confirms (.95) hemolysis.

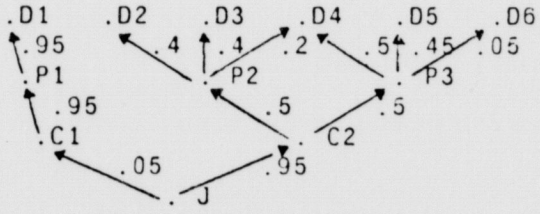


Figure 4. Part of AFFIRMING-tree in META-RULGEN, same notations as in Figure 3.

c. List of differential pairs (or groups) and mutually exclusive pairs (or groups) (see Figure 5): lists of incompatible diagnoses and findings. This list is constructed by recording pairs of individual facts that are incompatible (and often easily confused) with each other as reflected by the object-level rules. For example, hemolysis should be differentiated from Gilbert's disease (congenital conjugation defect), because they are similar in the aspect that urine bilirubin is negative.

```
((Hemolysis  Congenital-conjugation-defect)
 (Hepatitis  Calculous-jaundice)
 (Primary-biliary-cirrhosis  Calculous-jaundice)
 .......
 .......  )
```

Figure 5. List of differential or mutually exclusive pairs (or groups).

In these three structures, the current implementation allows no conjunction or disjunction in each node, i.e., each node represents a single fact.

3. Heuristics: the whole procedure for learning meta-rules (see section 2.3). The frequency table which is constructed by recording frequencies of attributes over the cases in the case library (see Figure 6) is also used to guide the program.

```
( (Acute-course  .5)
  (Malaise  .7)
  (Chills  .08)
  (Marked-body-weight-loss  .1)
  ......
  ......  )
```

Figure 6. Frequency Table. Each sublist contains an attribute with its frequency in the case library.

3.2 Algorithm

The search space of all possible meta-rules is roughly equal to the number of combinations of legal attribute-value pairs in the existing set of object rules. If all attributes were binary, this is the power set over n attributes. In the JAUNDICE program, n is 56 attributes in the 141 diagnostic rules (some attributes have binary values, and others have multiple values). The search is greatly constrained from the start by setting up the affirming tree and denying tree, which represent positively and negatively confirming links among attributes (and values) already noticed in the set of object rules.

There are two different parts of the algorithm, both of which are exercised. In our preliminary experiments we have taken the union of the two sets of meta-rules as the result. The algorithm is described for the separate approaches.

3.2.1 Approach from object rules

Form meta-rules on the basis of each individual object rule (of 141 diagnostic rules) as follows. Collect all object rules and form a set S. Take out the first element from S and do the following procedures; also delete this element from S.

- Step 1. Form the second part of the premise (conjunction of predicate with attribute- object- value triplet) on the basis of the premise part of the object rule. From the discussion in section 2.3, we note that a piece of evidence (premise in the object rule) is a plausible starting point to generate meta-rules.

- Step 2. For different conditions:

- a) Formation of pruning form meta-rules

    - i) If the object rule confirms some fact,by climbing the affirming tree, we know what other facts also are implied, with degrees of certainty calculated by propagating the uncertainty along the tree.[7] Thus, the third part of the premise will be those rule sets mentioning "presence of these facts", and these rule sets will be concluded to be useful in the action part of the meta-rule with degree of certainty calculated as described (see "Rule-in mechanism" in section 2.3.1).

    - ii) If the object rule disconfirms some fact, by climbing the denying tree, we know what other facts are also denied, with degrees of certainty calculated by propagating the uncertainty along the tree. Thus the third part of the premise will be those rule sets mentioning these facts,and these rule sets will be concluded to be useless with degree of certainty calculated as described. (see "Rule-out mechanism" in section 2.3.1)

    - Note that in both (i) and (ii), more than one meta-rule will be formed. There is a merging process in step 5.

---

[7] For example, if A implies B with degree of certainty .4 and B implies C with degree of certainty .6, then A implies C with degree of certainty .4 x .6 = .24

- b) Formation of reordering form By looking at the list of differential pairs, we know, for instance, Fact1 should be differentiated from Fact2. If the object rule confirms Fact1, then under the premise of this object rule, Fact1 should be pursued before Fact2. Thus, the third part of the premise will be rule sets mentioning Fact1, called Set1, or Fact2, called Set2, and it is concluded in the action part that Set1 should be invoked before Set2 with degree of certainty approximately equal to the degree of certainty of the object rule. Similarly, we can figure out the process if the object rule disconfirms Fact1 (see "Differential mechanism" in section 2.3.1).

- Step 3. Form the first part of the premise (the goal description) on the basis of the mentioned facts in the third part of the premise (the description of rule sets). For example, in Meta-rule079 (figure 2), the reordering of two facts: "cholestasis" and "parenchymal dysfunction" has to do with the conclusion of the subgoal "disease mechanism".

- Step 4. Calculate the Utility Value for each newly formed meta-rule, and filter out those with AUV being less than 1.

- Repeat the whole procedure until the set S is empty.

- Step 5. Meta-rules are further selected by RUV and verified by experimental simulations. Then, merge the meta-rules. If two meta-rules have the same first and second parts of the premise and their conclusions have no (or little) overlapping with respect to the concluded rule sets, then (a), add their descriptions of concluded rule sets to form a description of a new rule set and (b), add their conclusions (actions) to form a new conclusion (action). The Utility Value of the merged meta-rules is defined as the sum of the Utility Value of the individual rules before merging. Also, the redundant meta-rules are removed.

Example 1.

```
In the following descriptions, the notation
"-" means "absent" or "denied".

   A set of object-rules:

            .7
   R1: A1   -> -S1
            .6
   R2: A2   -> S2

   .......
```

```
  .......

A part of denying tree:
            1.
        -S1 -> -M1
            1.
        -S1 -> -M2

Form potential meta-rules on the basis of R1:
By propagation of implications:
            .7
       A1   -> -M1
            .7
 MR1: A1    -> Rules mentioning M1 are useless
Similarly,
            .7
 MR2: A1    -> Rules mentioning M2 are useless

(Note that "MR1" can be interpreted as:
 "If attribute A1 is present,
 then any rule mentioning presence
 of M1 will be useless,
 with degree of certainty .7",
 since M1 is denied by A1.)

Calculate the AUV as shown in section 4.

If both MR1 and MR2 are determined to
be retained after selections and their
conclusions are little overlapped,
then they are merged as:

      .7
 "A1 -> Rules mentioning M1 are useless
      .7
      -> Rules mentioning M2 are useless."

<Note:>
If a part of the differential list is:
   "((M1 M3) ......  )",
then a potential reordering meta-rule can be
formed:
            .7
 MR3: A1    -> Rules mentioning M3 DOBEFORE
               Rules mentioning M1,

since M1 is denied by A1.
```

3.2.2 Approach from attributes

Collect all parameters (attributes) to form a set S. Take out the first element and do the following procedures; also delete this element from S.

- Step 1. Form the first part of the premise. This is usually the main goal of the system, if there is only one main goal.

- Step 2. Form the second part of the premise by "presence of the parameter", and collect all object rules whose premise fails immediately because of "presence of the parameter". Thus, the third part will be those rules mentioning the presence of the parameter in their premise, and it is concluded in the action part that these rules will be useless with degree of certainty 1.

- Step 2'. Form the second part of the premise by "absence of the parameter ", and do the similar procedure as in Step 2. (Note: we try to form two meta-rules for each attribute with this approach.)

- Step 3. Calculate the Utility Value of each newly formed meta-rule, and filter out those with AUV being less than 1.

- Repeat the whole procedures until the set S is empty.

- Step 4. Meta-rules are further selected by RUV and verified by experimental simulations.

Example 2.

```
A set of object-rules:

  R1: A1 & A2 -> S1

  R2: A1 & A3 -> S2

  .......

  .......

Form potential meta-rules on the basis A1

  MR1: -A1 -> R1, R2, ..  useless.

Calculate AUV

........
```

.....

4 Preliminary Results

Sixty-three rules were created by Meta-Rulegen. About ten were formed by the approach from attributes and the remainder by the approach from the 141 rules in a preliminary version of the JAUNDICE system. About fifteen rules were reordering rules (all formed by the approach from object rules) and the remainder were pruning rules. (Figures 1 and 2 show one pruning rule and one reordering rule produced by the program.) We expand the rule sets into explicit lists of rules (Figure 7) to compute Utility Values of MR01 in figure 1 as follows.

AUV as: $\frac{20}{1} \times .97 \times 1 = 19.4$ and

$RUV = 19.4 \times \frac{100}{141} = 13.76$

```
Premise:
($AND (DIFFER LFT DOMINANT-BILIRUBIN INDIRECT))
Action:
(CONCLUDE (R79 R78 R61 R20 R19 R18 R17 R16 R13
           R12 R11 R10 R9 R8 R7 R6 R3 R109
           R117 R118)
 UTILITY NO 1.)
```

Figure 7. Name-referred form of meta-rule in Figure 1, in which the intensive definition of the concluded rule set is replaced by an extensive definition.

Table 1 shows the distribution of Utility Values among the 63 meta-rules. We informally confirmed that Utility Values are a reasonable standard, by looking at the medical significance of meta-rules (as found in the literature). We found that meta-rules with high Utility Values usually have high medical significance and conversely.

Table 1. Distribution of R.U.V. of 63 meta-rules created by META-RULEGEN.

	RUV			Total
	< 5	5 - 10	> 10	
Number of meta-rules	48	8	7	63

In addition, we performed a simple experiment to determine the effect of using some of these meta-rules in the JAUNDICE program. We selected 20 representative cases (non-randomly, but preserving the relative frequencies of diagnoses) among 72 cases collected from the literature, and ran them in batch mode in the JAUNDICE program. We measured the efficiency before and after incorporating different meta-rules. Table 2 shows selected portions of the outcome from which we see that the predicted Utility Value generally parallels the observed enhancement.

Table 2. Relationship between Utility Value and enhancement of system efficiency as determined from 20 cases run in JAUNDICE program (batch mode) with selected meta-rules.
(Lisp time: Lisp interpretation time)

	A.U.V.	R.U.V.	LISP time (sec.)	enhancement (percentage)
with MR01	19.4	13.76	271	15.3%
with MR34	13.07	9.27	293	8.4%
with MR40	11.5	8.1	298	6.9%
with MR07	1.94	1.38	330	-3.1%
with MR01 & MR34			256	20%
with MR01 & MR25 & MR37 & MR40 & MR45 & MR46			121	62%
without metarule			320	NA

The most important result revealed in Table 2 is the *additivity* of two non-overlapping (non-

overlapping of their concluded rule sets) meta-rules (e.g., MR01 & MR34). This important property can be proved formally (yet, we neglect the proof here), and it indicates that, by carefully selecting a set of useful meta-rules, overall system performance can be improved greatly (e.g., MR01 & MR26 & MR37 & MR40 & MR45 & MR46). However, there seems to be a limitation of the enhancement by combining several meta-rules.

Improved efficiency is only desirable in our system if there is no significant loss in performance. Thus, we compared the quality of the performance with and without meta-rules by asking whether the top disease diagnosis given by the system was the same as the expert's diagnosis and whether the associated degree of certainty was "close" (i.e., within .15) to the expert's confidence if the top diagnoses given by the system and the expert are matched.[8] The results show nearly complete coincidence: the only one imperfect match exists in an ambiguous case, in which two top diagnoses are given without meta-rules and only one top diagnosis is given with meta-rules. These results are expected because the meta-rules simply reorder the invocation of object-rules. If no certain conclusions are made, all object-rules will be activated anyway.

5 Conclusion

Intelligent control of inferences is important in knowledge-based systems for reasons of efficiency and human engineering, especially when knowledge bases become very large. In both cases, focusing the attention of the performance program can be accomplished by reordering and pruning elements of the knowledge base before invocation.

We have presented a general method for discovering meta-level knowledge that can be used to control inferences of an underlying performance program. The demonstration of the method is in terms of a rule-based representation of both object-level and meta-level knowledge, but we believe there is nothing specific to a rule-based representation in the method itself. The method depends only on representing the elements of the knowledge base in a network, with each node representing a fact and each link representing an inferential (or evidential) relationship between facts. In the case of a MYCIN-like rule base this means constructing three additional knowledge structures from

---

[8] In the SEEK program [6], the top model conclusion is compared with the expert's conclusion; however, the experts' confidence is not considered in comparison.

the rules themselves: an affirming tree, a denying tree and a table of differentials. These are explicit networks derived from the object-level rules showing facts that are positive evidence for other facts, negative evidence for other facts, or means of discriminating between two facts.

These three knowledge structures are analyzed in order to determine sets of nodes and links in the whole inference network that can be safely ignored in some contexts because they are seen to be irrelevant (i.e., false in those contexts). Similarly, the analysis can show parts of the network (sets of rules) that should be examined before other parts because that will increase efficiency.

From another viewpoint, the analysis mechanisms (Rule-in, Rule-out and Differential) are conditional search strategies because they help to select from a large stored knowledge source the best knowledge to apply. Meta-rules are just heuristics to select good and useful object rules, and Utility Value is one criterion for weeding out heuristics. Although we have designed our system to use degrees of certainty, an exact system without uncertainty can also be handled. It is an extreme case of a system with uncertainty in which uncertainty is quantized into two levels: True and False.

In summary, the concepts described in this paper can be easily extended to other AI systems because:

- Mechanisms such as, Rule-in and Rule-out, partition the reasoning network into smaller sets of knowledge and remove the useless ones. They can serve as the basis of forming both *control* and *search* strategy.

- Additional knowledge structures separate confirming and disconfirming links in the inference network. The nodes in these structures can be any fact in the world.

- The rules are not necessarily written in a MYCIN-like format. Moreover, knowledge can be represented in other ways, for instance, in a semantic net.

- The method used to handle uncertainty is not important. We may even use no uncertainty at all.

Acknowledgments

This work was supported in part by the Advanced Research Project Agency under contract DARPA N00039-83-C-0136 and National Institute of Health under grant NIH RR-00785-11.

References

1. Clancey, W. J., and Letsinger, R. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. Proceedings of the 7thIJCAI. Vancouver, B.C., IJCAI, 1981, pp. 829-836.

2. Davis, R. *Applications of meta-level knowledge to the construction, maintenance, and use of large knowledge bases.* Ph.D. Th., Computer Science Department, Stanford University, June 1976.

3. Davis, R., and Buchanan, B. G. Meta-level knowledge: Overview and applications. Proceedings of the 5thIJCAI. Cambridge, MA, IJCAI, August, 1977, pp. 920-927.

4. Miller, R A., Pople, H.E., Meyers, J D. "INTERNIST-1, an experimental computer-based diagnostic consultant for general internal medicine." *The New England Journal of Medicine 307* (1982), 468-476.

5. Patil, R. S., Szolovits, P., and Schwartz, W. B. Information acquisition in diagnosis. Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, Pa., AAAI, 1982, pp. 345-348.

6. Politakis, Peter G. *Using empirical analysis to refine expert system knowledge base.* Ph.D. Th., LCSR, Rutgers University, October 1982.

7. van Melle, W., *System aids in constructing consultation programs.* UMI Research Press, Ann Arbor, MI, 1980.

8. Wallis, J. W. and Shortliffe, E. H. "Explanation power for medical expert system: studies in representation of causal relationships for clinical consultations." *Methods Info. Med. 21* (1982), 127-136.