Knowledge in context:
A strategy for expert system maintenance

P. Compton and R. Jansen†

Garvan Institute of Medical Research, St. Vincent's Hospital
Sydney 2010, Australia
†CSIRO Division of Information Technology
PO Box 1599 Macquarie Centre
Sydney 2113, Australia

Abstract

Knowledge engineering, obtaining knowledge from experts and incorporating it into expert systems is difficult and time consuming. We suggest that these difficulties arise because experts never report on how they reach a decision, rather they justify why the decision is correct. These justifications vary markedly with the context in which they are required, but in context they are accurate and adequate; the difficulties in knowledge engineering arise from taking the justification out of context. We therefore hypothesise that knowledge engineering may be obviated, particularly in the long term maintenance of expert systems, if the rules experts provide are used in the context in which they are given. This paper describes work in progress to test this hypothesis.

Keywords and phrases: maintenance, context, expert systems, knowledge engineering, knowledge representation.

1. Introduction

There is little doubt that knowledge engineering, obtaining knowledge from experts and incorporating it into expert systems, is difficult and a major "bottleneck" in building expert systems. Initial problems arise in classifying and representing the knowledge domain; what data is the expert actually using to make his decisions and how many and what sort of decisions are being made? This classification problem can be difficult and must be faced before inductive learning techniques can be used (e.g.Quinlan *et al*. 1987), as well as in conventional knowledge engineering, where knowledge is elicited from the expert through interviews. With conventional knowledge engineering it can also be difficult to know how to elicit knowledge from the expert, but generally if suitable example data is available, knowledge can be readily obtained from experts. In our experience, the main problem with conventional knowledge engineering is that it is difficult to combine the fragments of knowledge obtained from the interviewing the expert into a consistent knowledge base. If the knowledge is obtained from experts in the form of production rules, *if certain conditions are fulfilled then conclude . . .,* then building an expert system requires combining such rules till the expert system has sufficient knowledge to emulate the expert's ability to make judgements in the knowledge domain. The knowledge engineering problem is that the rules added to the knowledge base generally conflict with the knowledge that is there already and the knowledge base must be manipulated until the conflicts disappear. Tools that check knowledge bases for subsumption, conflict etc. provide some assistance (Suwa *et al*. 1984), but they are restricted to checking the logical consistency of the knowledge base. It is quite possible to have rules which are logically consistent, firing and reaching opposing conclusions on the same data set

simply because the rules use disjoint subsets of the actual case data . Even when a logical inconsistency is flagged by a consistency checker, the knowledge engineer still has the job of tailoring the rules provided by the expert till they are consistent while still retaining the essence of what the expert said.

The conflicts that arise in adding to a knowledge base depend on the history of the knowledge base, what came first, what sorts of decisions where made etc. A decision to add a piece of new knowledge as a general, or specific, rule may have a major effect on how the knowledge base develops. Dependent on the decisions of the knowledge engineers, two expert systems for the same domain, developed with the same development tools, may result in quite different knowledge bases. Not only the knowledge engineer's decisions, but the development tools and inference strategy available will have a major impact on the resultant knowledge base (Bylander and Chandrasekaran 1986). As a result, it seems to us that knowledge base building is closely akin to a craft, with all the individuality and skills honed through practice that this implies. A good knowledge base has a similar beauty to other works of craftsmanship, and as with any work of art or craft any attempt to add to it or change it for a particular environment may spoil it. However, an expert system is a computer program intended for use in the real world where there will be demands for it to be changed and added to.

Expert systems have now gone beyond development to routine use and maintenance. During this maintenance phase new experts and knowledge engineers may be used, and even if the original personnel are available, their familiarity with the ways the knowledge was manipulated to fit into the knowledge base will have decreased. The history of the expert system, GARVAN-ES1, since it was introduced into routine use in mid 1984 (Horn *et al* 1985) provides an example of the maintenance that expert systems may require (Compton *et al* 1988). Since 1984 this system has been used continuously to provide clinical interpretations for reports from a diagnostic laboratory which measures thyroid hormone levels. The system chooses between some 60 interpretations and provides about 6,000 interpretations a year with each report checked and signed by an endocrinologist or senior biochemist. This checking provides a trigger for maintenance, since the experts signing the reports identify problems with the system. Since GARVAN-ES1's introduction the size of the rule base has more than doubled with half the increase due to the addition of new rules and half to the modification of existing rules. In this period acceptance of the reports by experts has increased from 95% to about 99.7%. (The relevance of these figures will be discussed further below.) This improvement has involved a considerable effort and it should be noted that the system has expanded only to more accurately carry out its original task. This contrasts with XCON, the best known example of long term maintenance, where the system has expanded due to new tasks and data as well as refinements in its knowledge (Bachant and McDermott 1984).

Knowledge engineering is difficult, and it seems reasonable to suggest that the difficulties will increase as expert systems move from the development phase into long term routine use. This transition stage into routine use probably categorises much current expert system development. The aim of this paper is to describe one aspect of work in progress to develop strategies to facilitate maintenance knowledge engineering. We assume that maintenance will be normally initiated by the failure of an expert system on a single case so that an expert will be the most appropriate source of generalised rules to handle further cases of this type. However, the initial knowledge base may be built by any means, for example, inductively from data bases of sample cases (Quinlan *et al* 1987).

It seems to be widely assumed that the reason why experts' rules generally have to be extensively manipulated before they can be added to an expert systems, is that it is difficult for experts to report accurately on how they reach decisions (e.g Waterman 1986). In our previous report on the problems of long term maintenance, we suggested that this was not an accurate model of the knowledge engineering process (Compton *et al* 1988).

> *The crucial observation was, that when experts were asked for their reasons for an interpretation, they appeared not to give the reasons why they reached the interpretation, but rather attempted to justify why the interpretation was in fact correct and this justification depended on the context.*

Although the expert might provide either simple heuristics or model based reasoning in justifying his interpretation, his choice seemed to rest on his assessment of the person he was communicating with, e.g. lab technician, scientist, clinician. The context also depended on the type of problem presented to the expert; e.g. with a single wrongly interpreted case, the rules would be quite different from when the expert was presented with a series of related cases. One simple way to observe this phenomenon is to obtain a rule for a specific case from an expert and add it to an expert system unedited. The new rule will virtually always conflict with or subsume other rules in the knowledge base. If cases normally interpreted by these other rules are then presented to the expert along with the original case a quite different rule for the original case will be provided.

Rather than conclude that experts are unable to report on their mental processes, such observations lead us to the hypothesis, that an expert identifies features in the data which distinguish the correct interpretation from other *likely* interpretations. The likely interpretations are of course those likely in the context. This is an example of Karl Popper's proposal that we never prove a hypothesis is correct, we only disprove alternate hypotheses (Popper 1963). Therefore, our reasons and justifications are only the reasons why the *alternate* hypotheses are wrong; they are not the reasons why every other possible hypothesis is wrong. We would suggest that this is the reason why experts' rules always appear too general when merged with other rules and applied to the real world. This hypothesis is not purely philosophical; studies of medical diagnosis are best explained by this method of reasoning (Griner *et al* 1981) and we have earlier suggested that such hypothetico-deductive reasoning explains how clinicians are able to cope with errors in laboratory analyses (Compton *et al* 1986).

Popper's hypothesis seems to model not only scientific research, where Popper has mainly applied it, and the communication of expert knowledge as we have proposed, but much of human communication. It seems to us that one of the central characteristics in any debate or argument is the willingness of the participants to change context at will. We are all familiar with the normal progression of an argument where neither side is ever wrong but the context in which one is right changes during the argument. It seems then that the ability to change context should be an essential component of expert system technology and in other work in progress, we are developing a knowledge base structure whereby the context in which the knowledge is viewed can be changed at will (Jansen and Compton 1988). This contrasts with the conventional way of examining a knowledge base by tracing the rules used in diagnosing test cases. The term context is widely used in expert system research but it seems to be mainly applied to fairly static, well defined contexts (e.g Shortliffe 1984). Our emphasis here is on the fluid and changing context in which an expert provides his justification for a decision.

Our central hypothesis for the study described here, is that the justification which an expert provides is highly accurate in context. Therefore if the reasons contained in the justification are used as rules in an expert system *in the same context,* they will be able to be used as provided, without further manipulation, thus making knowledge addition rapid and simple. This paper describes our progress towards testing this hypothesis. We have attempted to heed Buchanan's admonition that artificial intelligence research should produce measurable results (Buchanan 1988), hence the strategy we have adopted is to redevelop GARVAN-ES1 using rules in context and then attempt to measure the knowledge engineering problems and the performance of the resulting system against GARVAN-ES1. The technique we have adopted for capturing context is appropriate for laboratory report interpretation, but obviously other

methods will be more suitable for other domains; our purpose here is to discuss an example of a general principle.

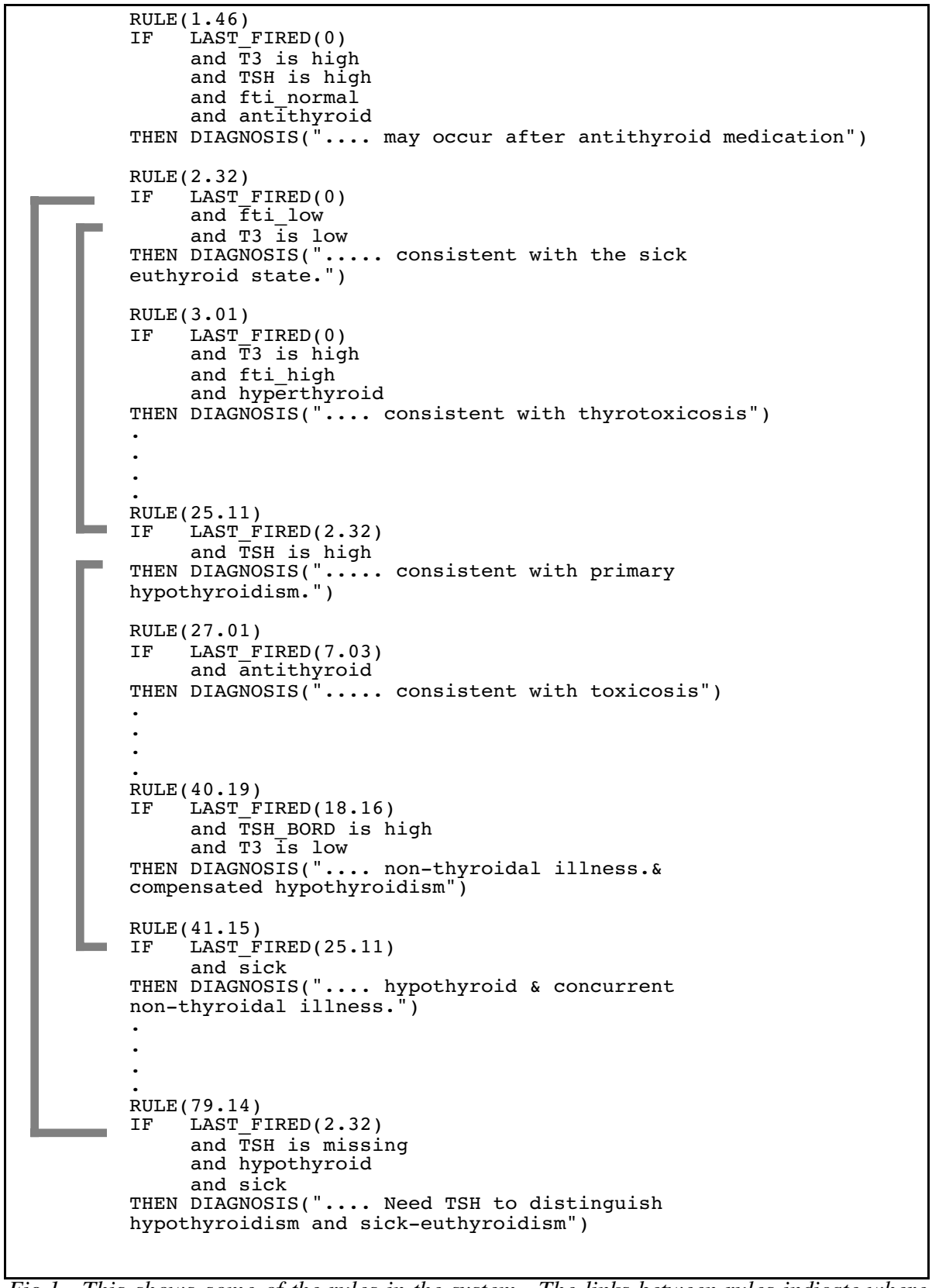
2. Methods

Context Strategy
The way GARVAN-ES1 is maintained is as follows: The expert checking and signing the reports produced by GARVAN-ES1 puts aside any reports with missing, incorrect or multiple clinical interpretations. A replacement report with the expert's comment is then immediately sent out to the referring clinician, while the original is kept for the knowledge engineer. If the error is not obvious the knowledge engineer consults an expert, and where necessary, the director of the Garvan Institute as the ultimate arbiter of GARVAN-ES1's knowledge. When the experts are consulted, they are presented with the report and any brief notes the referring doctor may have put on the request and asked why the old interpretation, or lack of an interpretation, is wrong and what rules should be used to arrive at the correct interpretation. The experts are not presented with any information about the rules the expert system used to arrive at the incorrect interpretation. We assume that the context in which the expert provides the new rules, or justification for the new interpretation cannot be exactly known, but will be largely determined by the material he is presented with. If the expert is giving a rule why a new interpretation is correct, this rule will be largely influenced by the context of the wrong interpretation he was presented with for this particular case and the features in the data he suspects may have lead to the wrong interpretation. In other words the expert's new rule is not a global new rule, but a rule to switch interpretations from the incorrect to the correct. Our previous study documents an example of this phenomenon (Compton *et al* 1988).

We attempt to capture this context by entering the expert's new rule directly as provided, but we include an IF LAST_FIRED(rule no.) condition (see examples in Fig 1). That is, the new rule will not fire on a case unless the old rule which produced the wrong interpretation has fired first. The new rule is not given the opportunity to fire until all the earlier rules have been tested. On the other hand if the expert system had not previously produced an interpretation for the case, then there are no indicators of the context available. However in this situation, the implicit context is that the expert system knew nothing about this type of case. Therefore the new rule should be used only if no other rules which could produce interpretations were able to fire. This new rule then contains an IF LAST_FIRED(0) condition and is not tested till all the earlier rules in the system have been tested.

To allow these strategies to work, each rule has only one opportunity to fire and the rules are tested in strict order from the oldest to the newest. Each new rule, whether it is a correction to an old rule or a rule for a previously uninterpreted case is added to the end of the list of rules. This ensures that the new rules are tested precisely in the context in which the expert provided them, that is the portion of the expert system that comes before this rule is exactly the same as the expert system which produced the interpretation shown to the expert earlier. Fig 1 shows some of the rules in the system while Fig 2 shows an alternate graphical representation of the structure. We adopted the further restriction that rules could not contain OR conditions to avoid problems where the correction to be made may only be relevant to part of the earlier rule.

Since this project is an attempt to simplify maintenance knowledge engineering, an important aspect of the approach is that rules are never corrected or changed, the corrections are contained in the rules added onto the end. In the formalism it is of course possible to attach a rule to explicitly contradict an earlier erroneous rule, but it was an interesting minor hypothesis to check whether in practice this would be needed, or whether there was always some truth in a rule, so that it was only necessary to narrow it by later rules. We hypothesise that there is generally some truth in what experts and the rest of us have to say, so knowledge should never need to be thrown away completely. The problems arise when we apply our insights out of context. The other important aspect of the approach is that the rules are added exactly as the expert provided them; they are not manipulated to minimise conflicts within the knowledge base.



```
RULE(1.46)
IF   LAST_FIRED(0)
     and T3 is high
     and TSH is high
     and fti_normal
     and antithyroid
THEN DIAGNOSIS(".... may occur after antithyroid medication")

RULE(2.32)
IF   LAST_FIRED(0)
     and fti_low
     and T3 is low
THEN DIAGNOSIS("..... consistent with the sick
euthyroid state.")

RULE(3.01)
IF   LAST_FIRED(0)
     and T3 is high
     and fti_high
     and hyperthyroid
THEN DIAGNOSIS(".... consistent with thyrotoxicosis")
.
.
.
.
RULE(25.11)
IF   LAST_FIRED(2.32)
     and TSH is high
THEN DIAGNOSIS("..... consistent with primary
hypothyroidism.")

RULE(27.01)
IF   LAST_FIRED(7.03)
     and antithyroid
THEN DIAGNOSIS("..... consistent with toxicosis")
.
.
.
.
RULE(40.19)
IF   LAST_FIRED(18.16)
     and TSH_BORD is high
     and T3 is low
THEN DIAGNOSIS(".... non-thyroidal illness.&
compensated hypothyroidism")

RULE(41.15)
IF   LAST_FIRED(25.11)
     and sick
THEN DIAGNOSIS(".... hypothyroid & concurrent
non-thyroidal illness.")
.
.
.
.
RULE(79.14)
IF   LAST_FIRED(2.32)
     and TSH is missing
     and hypothyroid
     and sick
THEN DIAGNOSIS(".... Need TSH to distinguish
hypothyroidism and sick-euthyroidism")
```

Fig 1. This shows some of the rules in the system. The links between rules indicate where rules were added to correct an interpretation given by an earlier rule. The rules are stored as a single long list, with new rules added to the end of the list. The gaps in the table indicate that rules have been omitted from this extract. The decimal rule numbering system encodes extra information not relevant here.

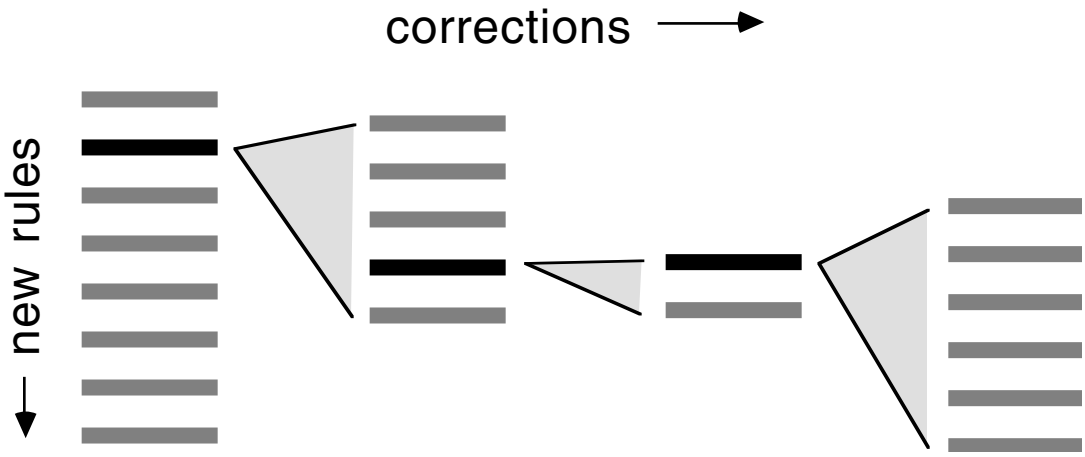


Fig 2 This figure show an alternative representation of the structure of the system. Each horizontal bar represents a rule. New rules are added to the bottom of each stack of rules. The first stack are rules which can only fire if no previous rule has fired; they are checked to see if any can fire from top to bottom. As soon as a rule fires (highlighted) the only rules which can fire are those which have that rule as a precondition. Any new correction rules with the same precondition are put on the bottom of this stack. If none of the correction rules can fire, the interpretation is taken from the precondition rule; i.e. the last rule which fires gives the interpretation. (Note: this diagram displays the logical connections between rules; there is only one stack of rules as this logical structure is achieved by coding the rules as a single long list as shown in Fig 1)

Test environment

We have proposed this strategy for the maintenance phase of an expert system project but we have tested it here by attempting to rebuild GARVAN-ES1 from the ground up using this "ripple down" rule approach. The knowledge engineering was entirely case driven as we entered rules only as required to interpret specific cases, from the first rule onwards. There was no attempt to put the most general rules first; rules were only added as required for case interpretation since we wanted to make the test environment as challenging as possible. It is important to note however that this did not mean that specific rules were added. The system uses rules that are obtained directly from experts and these are always general rules (Waterman 1986), but we made no attempt to put the most general of these general rules first. The exceptions to this approach were that we used the set of rules that initially classify the assay results as low, normal, high etc directly from GARVAN-ES1. We similarly used the rules which classify the referring doctor's comments, the type of referral, and the patient's age and sex from GARVAN-ES1. Although these rules have grown during maintenance, particularly the comment rules, there have been few knowledge engineering problems as the changes were generally simple additional rules which identified strings in clinical comments as indicating the type of comment being made. All the rules which used such comment classes in conjunction with other data to reach an interpretation were rebuilt.

The cases used to rebuild the system were the cases in the "cornerstone case" data base plus cases from the Garvan archives. The "cornerstone case" data base consists of those cases which have required a change in the rules of GARVAN-ES1. Each time the rules in GARVAN-ES1 are modified all the cornerstone cases are retested to ensure that the knowledge addition has been incremental, and that the knowledge base had not been corrupted by the changes (Horn *et al* 1985, Compton *et al* 1988). We started with the cornerstone cases to speed development, in that we would fairly quickly cover cases which had been important in the development of GARVAN-ES1. On the other hand many of these cases are unusual, meaning that we would be starting from particular rather than general rules. The data base of cornerstone cases contained 291 cases at this stage. When the system interpreted the cornerstone cases correctly we tested it against 9514 cases we had in a larger data base. We then proceeded to work through the failures on these cases in chronological order to simulate

the actual chronological maintenance process. These cases were actual patient records covering the years 1984 to 1986, with gaps in the data being due to technical difficulties in retrieving the archives rather than any selection process. Any cases which required a change in the rules were copied into the cornerstone case data base.

We did not use an expert to provide the interpretations for the cases, but took the interpretations provided by GARVAN-ES1 as our reference for correct interpretation since the comments produced by GARVAN-ES1 have an apparent 99.7% acceptance by experts (see below). As well as avoiding demands on experts this then gave us opportunities to measure the performance of the new system. It should be noted that since all the cases used were actual patient records, experts had previously checked and signed reports for these cases including GARVAN-ES1 interpretations. Essentially then our references for correct interpretation are expert approved interpretations with the proviso that GARVAN-ES1 has undergone maintenance (Compton *et al* 1988) since the original reports were issued. P.Compton has been responsible for most of the maintenance for GARVAN-ES1 and has had considerable experience in laboratory endocrinology and so was able to function as the expert providing the rules to give the interpretations required by GARVAN-ES1.

For the first case a rule was simply entered into the empty expert system, but from then on each new case was run on the ripple down rule expert system, as it had developed thus far. The interpretation, or lack of interpretation, was then compared to the interpretation made by GARVAN-ES1 and a new rule added to the system as required. As stated above no rules in the system were changed, rules were only added. Also to test the approach, we only periodically checked whether the new rules had affected the performance of the system by incorrectly interpreting cases previously interpreted correctly. The occasions when data base of cornerstone cases was checked and further rules added to restore previously correct interpretations are shown in Fig 4. This approach contrasts strongly with the imperative requirement to carefully check the effect of all rules added to more conventional knowledge bases.

Our initial intention was to enter the new rules without any reference even to the rule which produced the wrong interpretation. In fact it was necessary to look at both the rule which had fired incorrectly for the new case and the case which had required this earlier rule to be added to the system. The main reason for this was because we were in fact trying to capture an expert system's rather than a human's knowledge. We found that some of GARVAN-ES1's interpretations of the 9514 cases were obviously due to uncorrected aspects of its knowledge, buried in the complexity of the rules and picked up only by rare data profiles. We have previously noted that the checking process may allow clinically unimportant errors in interpretation to be retained by the system[1]. It was often difficult to think up rules to track these vagaries without inspection of the earlier case and rule. However all of these interpretations were captured in the ripple down system so that its performance could be accurately quantified against GARVAN-ES1. A second problem was that since an experienced knowledge engineer rather than an expert provided the rules, some of the rules were less than optimal and it was occasionally difficult to know what new rule needed to be introduced without looking at the rule which had lead to the wrong interpretation. This problem occurred even though it was a single person who composed all the rules over a period of a few months. These however are very minor constraints on the freedom with which rules are added when compared to conventional rule addition. In conventional rule addition the performance of the whole system must be checked with little way of predicting which rules will be affected by the rule addition. We are here talking about simple addition of rules where the parent rule and parent case may need to be looked at to compose the new rule. It was also

[1] In studies of inductive learning with the program C4 (Quinlan et al 1987) it has emerged that in up to 2% of reports, experts may formulate a more sophisticated interpretation if challenged to provide their most comprehensive analysis. Since experts have to go to some trouble to put aside a report it seems that they do not do so if a more refined or slightly different interpretation will not be of diagnostic importance to the referring clinicians.

necessary to test the new rule since simple clerical blunders in the entry of the rule occurred (as always). At no stage however were the actual rules used by GARVAN-ES1 used to provide information. In any case, as the samples in our earlier report (Compton *et al* 1988) suggest this would not be helpful because of the complexity of some of the rules that has resulted from maintenance.

Implementation
The shell is an adaptation of the original GARVAN-ES1 written in C and which runs on a PDP 11/73 under the TSX Plus operating system. A preprocessor translates the English like rules to C code. As the system is restricted to a 64K job space overlaying is required, but the 9514 cases can still be checked by both versions of the GARVAN-ES1 on a moderately loaded system in under six hours, about one second per interpretation. The main tools used were programs to compare the interpretations provided by the two expert system on cases in the data bases. These programs could be run in batch mode or interactively where the ripple down rules which had fired for a case could be viewed as well as any intermediate facts that had been asserted and the case data itself. The new rules were composed and entered manually.

3. Results

This study is not yet complete, but there are sufficient data to indicate that the approach proposed does facilitate some aspects of knowledge engineering. One important result is the speed with which ripple down rules can be added. One can add ten rules per hour without much difficulty and sixty rules were added in one eight hour period, where rule production was logged. The problems in adding the rules were the typical problems of data entry, fatigue, clerical errors etc. In contrast, with GARVAN-ES1, rule addition often takes half a day per rule. With a conventional system the problems are that the existing knowledge in the system must be explored and tested; are there rules in the system which could be generalised to cover the new case; will a new rule conflict with or subsume other rules; if a rule is narrowed will it still interpret the cases it should? These questions apply not just to the logic of the rules, an area where some sort of consistency check may be of assistance, but more importantly to the performance of the rules. Does the manipulation of the expert's knowledge required result in rules which are unrecognisable to the expert? All these question must be answered each time the knowledge is changed in a conventional system. As discussed above tools to check the logical consistency of a knowledge base can assist the knowledge engineer

We proceeded by first of all developing rules to cover the 291 cornerstone cases and then developed rules for the cases in the larger data base which were not yet correctly interpreted. We considered these cases a thousand at a time to have some index of development. To date we have completed development for 3000 of these cases. The accuracy of the rules is shown in Fig 3. The rules developed from the cornerstone cases alone correctly interpreted 88% of the larger data base, however after testing 3000 cases in the larger data base and adding the required rules, the error rate had only dropped to 6% if the remaining unused 6514 cases are tested, or 4.4% if all 9514 cases were considered. The test of all the data is of minor importance, but is included because as Fig 6 indicates errors may be introduced undetected for cases previously interpreted correctly.

Figure 4 shows the number of rules that have been added to date, a total of 347. In the corresponding version of GARVAN-ES1 there are 183 rules, with these numbers excluding the classification rules that are common to both systems. GARVAN-ES1 contains OR conditions which are excluded from the ripple down system. In other work, storing the knowledge in GARVAN-ES1 in an expanded data dictionary (Jansen and Compton 1988), the disjoint normal form has been used which increases GARVAN-ES1 to about 600 rules. It appears that the two systems will be of comparable size. Of the 183 rules in GARVAN-ES1, 49 are used to produce intermediate assertions used by later rules. In the ripple down rule

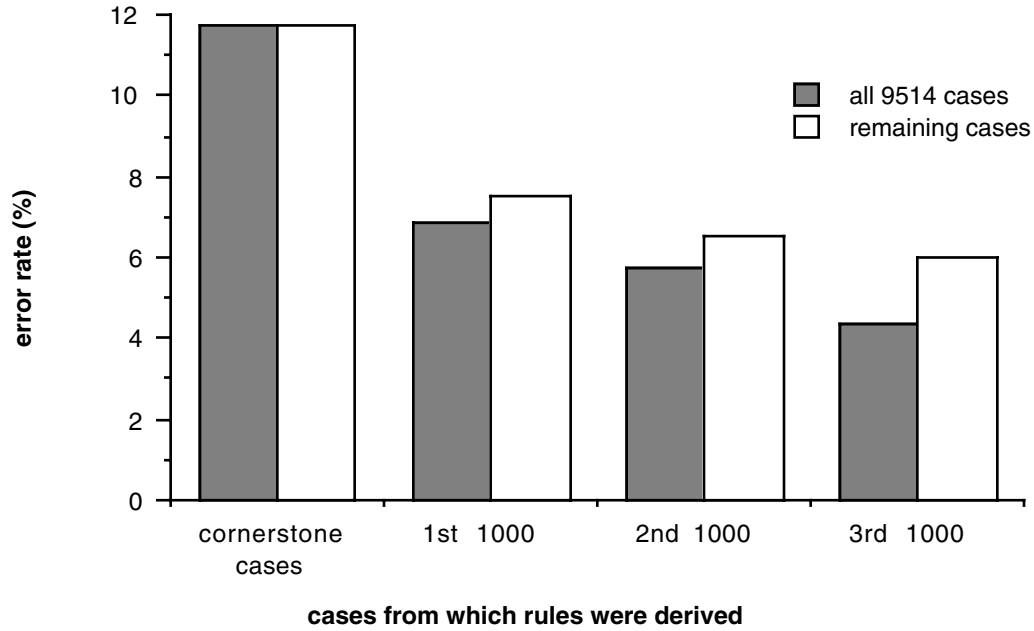


Fig 3. This shows the percentage of incorrect interpretations in the large data base at various stages of the ripple down rule development., after rules had been developed to handle the cornerstone cases, the first thousand cases in the large data base and so on The error rates are given for the whole data base, as well as for the cases which have not yet been used for new rule development.

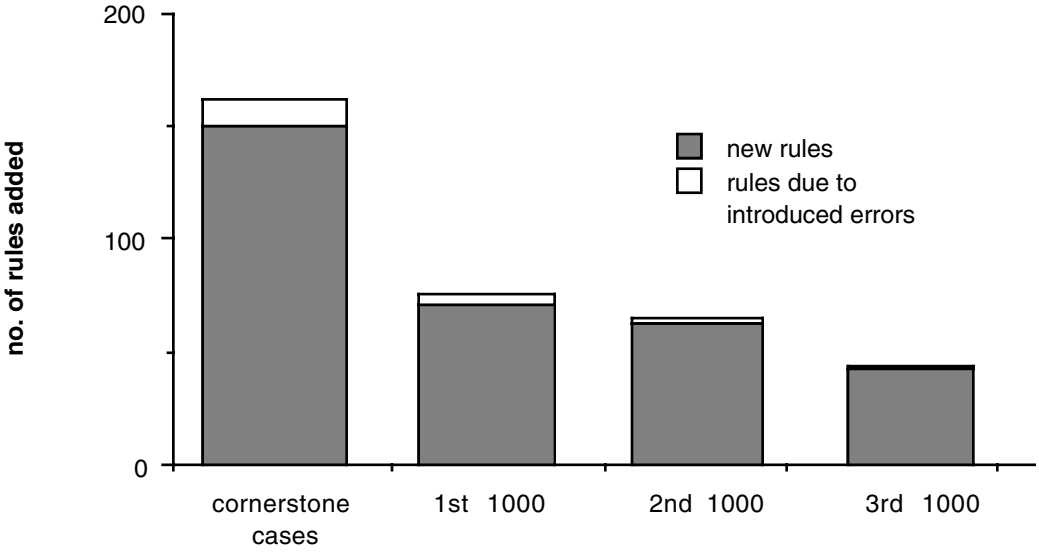


*Fig 4. This histogram shows the number of rules that were added to cope with the erroneous interpretations in each group of cases, the cornerstone cases, the 1st 1000 etc. Cases from the large data base which required new rules were added to the cornerstone case data base. At the **end** of each section of additions, all the cornerstone cases where checked and further rules added to handle any further corrections required because of now incorrect interpretations of cornerstone cases. The numbers of these rules are shown in the lightly stippled sections of the histograms. The heavily stippled areas also indicate the number of cases added to the cornerstone case data base after the first 291.*

system, apart from the initial classifications common to both systems, no intermediate assertions are made, only interpretations which are corrected by later rules.

The ripple down system needed only 161 rules to be able to interpret all of the 291 cases initially in the cornerstone case data base, and which in essence traced the history of GARVAN-ES1 maintenance. However in the test on the larger data base, these rules were not able to interpret all the cases in the same way as GARVAN-ES1, as shown in Fig 3, indicating that different attempts to build rules for the same task will capture different features in the data. Fig 4 also indicates the numbers of cases added to the cornerstone case data base for each thousand cases tested from the larger data base. There are now 467 cornerstone cases. Fig 4 also shows a slightly stronger improvement in accuracy as the system develops than Fig 3. Fig 4 shows there were 71 cornerstone cases, or new rules, for the first 1000 cases, but only 44 for the third thousand cases. This contrasts with the error improvement shown in Fig 3 which goes from 7.5% to 6% as measured on the remaining cases. There may well be repeated examples of errors in the remaining data set, but we won't know till the development is complete.

The most important data in Fig 4 are the further corrections that were required to be made after rules had been entered. Of the 347 rules entered only 20 were due to the cases in the cornerstone case data base being picked up incorrectly by later rules. Only one in every sixteen rules created problems. This is in complete contrast to conventional rule addition. For example, an audit log was kept of the rule changes in GARVAN-ES1 required by the last 12 cases added to the data base of cornerstone cases before this project commenced. It took 31 attempts to introduce appropriate changes for these 12 cases, with one case requiring 5 attempts at introducing appropriate changes and another case 4. For 2 of the 12 cases the first draft changes caused no further problems. Often a change would cause a number of cornerstone cases to fail, but normally no more than 3 or 4. That is, initial attempts at putting in new rules or changing rules for 12 cases caused 19 sets of unexpected changes to the performance of the system when tested against the cornerstone cases. This is the main area of contrast between conventional and ripple down knowledge engineering. Rule changes to conventional systems have to be carefully engineered to avoid corrupting existing knowledge while in comparison, ripple down rule addition has virtually no effect on existing knowledge The knowledge in GARVAN-ES1 is rather flat so we assume changes to it have far less impact than with a deep reasoning system, so the contrast of maintenance on a deep reasoning system with the ripple down system is likely to be even greater.

Fig 5 gives some indication of the "usefulness" of each rule by showing the number of cases effectively handled per rule. The high efficiency of the rules added for the cornerstone cases, is somewhat misleading as no interpretation is given for normal results (not necessarily meaning all results are within their normal ranges), and these cover about 70% of the cases seen. The percentage figures give a somewhat truer estimate of the potency of the rules. The figure of 2 cases per rule for the third thousand suggests about another 200 rules are required to complete the system. Fig 6 shows important data because it indicates how even with a data base of cornerstone cases, changes to a knowledge base can cause other unanticipated changes in the knowledge. These are changes that were only picked up by testing all the previously interpreted cases. Some of the errors introduced by the rules added for the first thousand cases were corrected by rules introduced by the requirements of the second thousand, but these in their turn introduced other errors in the first thousand as well as the second thousand. None of these introduced errors have been corrected except when they are corrected by the introduction of a rule to deal with a later case. We did this to simulate real life maintenance, where this sort of error would be expected to occur.

The minor hypothesis that no rules would need to be completely negated by later rules was also supported. There was always some context in which the rule was true. It seems highly unlikely that an expert will provide a rule which is completely wrong under all circumstances; the more likely problems are excessive generalisation, or attention to peripheral data. In both these cases there is still a context in which the rule is right, and ripple down rules provide a means of narrowing the context without having to change the original rule. We include this

minor result since some find it contrary to their expectations of what is involved in knowledge maintenance.

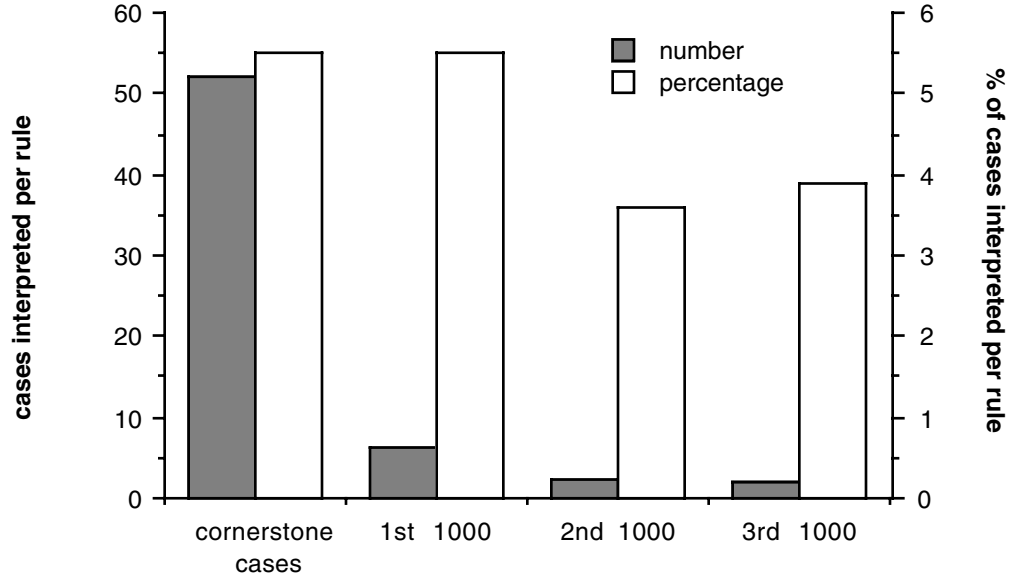


Fig 5. This indicates the number of cases interpreted per rule added. The number is calculated as the number of cases, which are interpreted per rule. The number of cases interpreted per rule is also expressed as a percentage of the the number of cases awaiting interpretation when the rule was added.

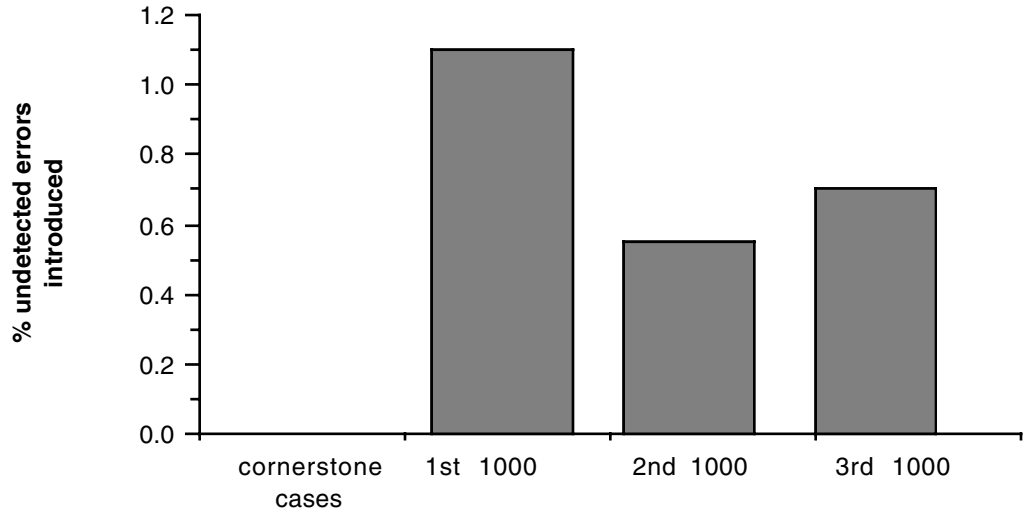


Fig 6 . After rules were composed where necessary for each thousand cases, all the cases were rechecked. This figure shows the errors introduced into cases which were initially interpreted correctly. The cases retested for the 2nd and 3rd thousand columns include all of the two thousand and three thousand cases tested initially

4. Discussion

The most important conclusion that can be made from this study so far, is that if a ripple down rule structure is used, rules can be added to an expert system directly as obtained from an expert without further manipulation. Rules added to this structure do far less to corrupt the the existing knowledge in an expert system than with a conventionally structured expert system. Secondly, rules added in this fashion can be added far more rapidly than with a conventional system, so that requirements for maintenance become minor irritations rather than major events requiring hours if not days of work per addition to the knowledge base. The potential difficulty with such a system is that later knowledge added to the system is far more restricted than in a conventional expert system and so may have to multiplied throughout the system. In this study we have attempted to build an expert system completely from ripple down rules to evaluate the method. However our proposal is that the method should be used for the maintenance phase, not for building all of an expert system. In the maintenance phase the emphasis should not be on new general rules, but small corrections and additions to a mature system where the priority is to handle the correction as easily as possible. The other problem where a system is required to handle major new areas of knowledge needs to be handled by tools like the "knowledge dictionary", an extended data dictionary that we have under development (Jansen and Compton 1988)

Although one would expect problems to arise with a ripple down system because the same knowledge may have to be repeated in rules throughout the system, the counter balance is that experts' rules can be entered into the system in their most general form. In a conventional expert system, as the knowledge becomes more complex, it can be very tempting to add a highly specific rule, far more specific than as supplied by the expert, to avoid interactions with the other knowledge. With a ripple down system, a rule can be used in its most general form as supplied by the expert since the structure controls where this general rule is applied, but this rule may have to be reproduced elsewhere. The reason we have attempted to rebuild GARVAN-ES1 completely with ripple down rules is to measure the cost of this repetition. This work is not yet complete but it seems that the resulting system will probably be not very much bigger than GARVAN-ES1. In its disjoint normal form GARVAN-ES1 has 651 rules where the ripple down system has 347 rules and another 415 cases still to interpret.. Although the new system will be bigger, it takes less attempts per rule addition and rule addition is far faster; in contrast to a conventional system, rule addition does not become any more difficult as the system grows. We would note again that we made no attempt to put the most general rules first in order to provide a more stringent test of this approach. In practise one would of course attempt to put the most general rules first and in fact as we have suggested it may be preferable to use a conventional approach for the core knowledge and move to ripple down rules once the core knowledge is in place. We also note that although ripple down rules may result in redundancy, this redundancy itself may provide a reliable resource for automatic generalisation of knowledge. The expert's rules are captured as expressed and the possible generalities in them can be extracted later, rather than the knowledge engineer trying to decide what are appropriate generalisations, and thereby irretrievably burying the expert's knowledge.

We propose to test other strategies for capturing context, for example using an IF LAST_INTERPRETATION(interpretation) condition. This would be more general than the LAST_FIRED(rule no) condition, which catches some of the data profile as well as the interpretation being made as part of the context. Since this strategy would be more general it would probably introduce more knowledge engineering problems, but this remains to be assessed. We also propose to repeat some of the work here using an actual expert rather than an experienced knowledge engineer doubling as an expert. It will be interesting to see if the expert's rules are more general or more restricted. The LAST_FIRED() strategy has been useful for the GARVAN-ES1 domain but obviously other techniques will be required to apply experts' knowledge in context in other domains. For example Prolog "worlds" (Menzies and Worrall 1987) may provide a basis for a range of flexible strategies for capturing an expert's knowledge in context for other domains.

This system requires no conflict resolution strategy or probabilistic reasoning. It has always been a problem with using probability that experts don't seem to use probability in the rules they provide. We suggest that this is because experts provide a rule for a context; probability and conflict resolution only come in as attempts to apply the rule in the wider world. Ripple down rules attempt to retain the context in which the rule was provided. Conflict resolution strategies are attempts to enhance the inferencing power of a system, to do the job when the knowledge isn't available. The "knowledge principle" (Hayes-Roth *et al* 1983) proposes that knowledge is power, and that artificial intelligence programs work because they know a lot about the world rather than because they have clever reasoning methods. Ripple down rules are an attempt to use knowledge rather than reasoning. The significance of the method is that it recognises that expert knowledge, as it is expressed, is a justification of the expert's judgement in a specific context, and it attempts to ensure that the justification will only be used as a rule in the same context.

An important consequence of using knowledge as expressed by the expert, is that a trace of the rules which have fired in providing an interpretation, provides a far more intelligible explanation of the systems reasoning than with a conventional rule trace. The ripple down rule trace shows rules exactly as expressed by experts apart from some computer type names (eg Fig 1). In contrast conventional rules may be quite opaque, because of the tailoring to the other knowledge required and the accretions over time (eg Compton *et al* 1988). Secondly, the rule trace shows not only the rules that fired but the history of the corrections and additions to the knowledge, as applicable to the case, and so has more potential as an educational tool, than a conventional rule trace.

The order of the rules is all important in a ripple down rule system. One expects in an expert system that the knowledge be separate from the inference procedure, and this was a major issue in the emergence of expert systems as different from conventional programs. But the hidden agenda in expert system building is the enormous amount of work the expert system builder puts into crafting a coherent and consistent knowledge base *which works*. The rules may look like knowledge independent of an inference method, but in fact as any experienced knowledge engineer knows the rules have been extensively massaged till they work correctly with the inference engine. As Bylander and Chandrasekaran (1986) have pointed out there is no such thing as knowledge independent of an inference strategy. It is this hidden agenda that makes an expert system like a work of craft or art, as we mentioned earlier; but most of the craftsmanship is not contained in the rules, but in their invisible relationships. As such the knowledge base, like any work of art cannot be added to without possibly destroying it; expert systems become unmaintainable. The approach we have suggested is to have the structure and the relationships completely explicit, so knowledge can be crudely added to the end at any time. Because the structure is so visible with ripple down rules it does not constrain the development and becomes essentially invisible. On the contrary in a conventional expert system where the structure is hidden, the knowledge engineer becomes not the blindfolded man trying to tell what an elephant is, but the blindfolded man trying to fill in the parts of a jigsaw of an elephant.

Ripple down rules seem to relate to other strategies to build decision trees such as ID3 (Quinlan *et al* 1987) but there are important differences. ID3 and its descendants attempt to extract decision trees or rules from data and the more representatives there are of any particular data class, the better they work and they identify the statistically most important features in the data. The ripple down rules approach on the other hand attempts to capture rules exactly as expressed by experts and use them in the same restricted context in which they were obtained to avoid corrupting the rest of the knowledge base. The strength of inductive learning is in finding the most general rules for a knowledge base, while the strength of ripple down rules is handling the minor rules and later additions. One can attempt to use either technology exclusively, but perhaps the ideal solution is to use each technology at the stage of the project where it is most suited.

5. Conclusions

Our conclusion from this work in progress is, that to facilitate maintenance, corrections to a knowledge base should only be used in the context in which they were supplied. In this way knowledge can be added directly as supplied by the expert, resulting in far simpler addition and far fewer unexpected affects.

Acknowledgements

This work is part of a collaborative project between the Garvan Institute of Medical Research and the CSIRO Division of Information Technology. We wish to thank our colleagues in both institutions for their support, particularly Les Lazarus of the Garvan and Bob Colomb of CSIRO. We are indebted to Nigel Morrison of the Garvan who first suggested that we should hang new rules from the end of an expert system. We also wish to thank Kim Horn and Ross Quinlan, without whom there wouldn't be a GARVAN-ES1 to base this study on, and the Garvan staff who have provided endocrine expertise over the years.

References

Bachant, J. McDermott, J. "R1 revisited: four years in the trenches", The AI Magazine (Fall 1984; Fall), pp.21-32

Buchanan, B.G. "What do expert systems offer the science of artificial intelligence", Proceedings of the fourth Australian Conference on Applications of Expert Systems, (1988), pp.1-30

Bylander, T. Chandrasekaran, B. "Generic tasks for knowledge-based reasoning: the 'right' level of abstraction for knowledge acquisition", Proceedings of the knowledge-based systems workshop, Banff, (1986), pp.7.0-7.13

Compton, P. Horn, K. Quinlan, R. Lazarus, L. "Maintaining an expert system". Proceedings of the fourth Australian Conference on Applications of Expert Systems, (1988), pp.110-129

Compton, P.J. Stuart, M.C. Lazarus, L. "Error in laboratory reference limits as shown in a collaborative quality assurance program", Clin Chem, Vol 32, (1986), pp.845-9

Hayes-Roth, F. Waterman, D.A. Lenat, D. "An overview of expert systems", In: *Building expert systems*. eds., Hayes-Roth, F. Waterman, D.A. Lenat, D. (1983), pp. 3-29

Griner, P.F. Mayewski, R.J. Mushlin, A.I. Greenland, P. " Selection and interpretation of diagnostic tests and procedures". Ann Intern Med, Vol 944, (1981), pp.553-92

Horn, K. Compton, P.J. Lazarus, L. Quinlan, J.R. "An expert system for the interpretation of thyroid assays in a clinical laboratory", Aust Comp J, Vol 17, (1985), pp.7-11.

Jansen, R. Compton, P. "The knowledge dictionary: an application of software engineering techniques to the design and maintenance of expert systems", Proceedings of the AAAI'88, workshop on the integration of knowledge acquisition and performance systems, (1988), in press

Jansen, R. Compton, P. "The knowledge dictionary, a relational tool for the maintenance of expert systems", Proceedings of FGCS'88, (1988) in press

Menzies, T. Worrall, C. "Worlds in Prolog" Proceedings of AI'87 (1987) pp383-393

Popper KR. (1963) *Conjectures and refutations* Routledge and Kegan Paul Ltd., London

Quinlan, J.R. Compton, P.J. Horn,K.A. Lazarus, L. "Inductive knowledge acquisition: a case study", In: *Applications of Expert Systems*. ed., Quinlan JR, Addison Wesley, London (1987), pp. 159-73.

Shortliffe EH. "Details of the consultation system" In: *Rule-based Expert Systems,* eds., Buchanan BG and Shortliffe EH. Addison Wesley, Readingg Mass, (1984), pp.78-132.

Suwa M, Scott AC, Shortliffe EH. "Completeness and consistency in a rule-based system" In: *Rule-based Expert Systems,* eds., Buchanan BG and Shortliffe EH. Addison Wesley, Readingg Mass, (1984), pp.159-70.

Waterman DA. (1986) *A guide to expert systems* Addison Wesley, Reading, Mass.