Cornerstone Cases in A Dictionary Approach to Rule Maintenance

P.Compton†, W.Yang†, M.Lee* , B.Jansen*

† Dept. of Computer Science, University of New South Wales,
PO Box 1 Kensignton 2033, Australia

* CSIRO Division of Information Technology,
PO Box 1599 Macquarie Centre
North Ryde 2113, Australia

Abstract
Maintenance problems with knowledge bases occur because experts always provide their knowledge in a particular context. This context is largely determined by the case which prompted the change to the knowledge base. Since the case is an important determinant of the changes in knowledge it should be included as an integral part of the knowledge base so that context can be recalled to help determine any further maintenance changes. It further seems possible that if cases are included as part of the knowledge base, it is possible to greatly simplify maintenance. Maintenance may be able to be simplified to the extent that the expert is only required to select form a list of conditions to produce a rule which is guaranteed to be correct from and engineering perspective. The list of conditions to be picked from can be determined by the differences between the cases for which maintenance is required and previous cases for which maintenance has been required.

Introduction

Knowledge maintenance

Knowledge maintenance is a major problem with expert systems. The studies with R1/XCON (van de Brug, Bachant et al. 1986) indicate the problems in a domain where the scope of the problem is constantly changing as Digital Equipment introduces new computers into its product range. The studies with GARVAN-ES1, an expert system which provides automatic clinical interpretations for pathology reports indicate the problems in a domain where the scope of the problem remained constant over the 5 years the system was in use (Compton, Horn et al. 1989; Compton and Jansen 1990). The system was constantly maintained and doubled in size over the period taking its accuracy, or rather the acceptance of its interpretations by experts, from 96% to 99.7%. Note that the system was not withdrawn from use because of any failure but because a transfer of the assay service to another department of the hospital where the knowledge base has been redeveloped as part of a larger and more comprehensive knowledge base (Compton, Srinivasan et al. 1991).

The knowledge maintenance for GARVAN-ES1 was carried out manually with the only tools being a text editor and a database of cornerstone cases (Horn, Compton et al. 1985). These cornerstone cases were cases for which the knowledge had previously been modified. Any cases which required a change in the rules were added to this data base and every time any further changes were made, all the cornerstone cases were tested to see that they still had the correct interpretation; i.e. the change to the knowledge base was incremental. It is not unusual to have a data base of test cases to test changes to a knowledge base (Buchanan, Barstow et al. 1983) but the data base here is unusual in that it consists only of cases which have been responsible for earlier changes to the rules - hence the name cornerstone cases. A number of strategies evolved in using this data base which will be of relevance here. Essentially all problem with the knowledge base can be reduced to the two simple cases of:

1. If a rule is firing inappropriately it has to be narrowed by adding further conditions so that it will not fire on the case in hand. This may be the rule that actually produces the interpretation but it also may be an earlier rule in the inference chain that produces an intermediate assertion used by the later rule. Either way the scope of a rule is narrowed by adding further conditions to it. The cornerstone cases must then be tested to see that all cases that previously satisfied the rule still satisfy it.

2. The second situation is that a rule has to be expanded to encompass a case that should satisfy it but doesn't. One expands a rule by removing a conditions. Alternatively a new rule has to be made up to encompass the case. This is the more difficult situation, because in the case of expanding the rule one first has to find the rule which should be expanded. One can think of a number of ways to find the best candidates to change. In practice we adopted the strategy of making up a very general new rule for the case and then testing the cornerstone

cases against this new rule. The cases that caused the new rule to fire were then examined to see if one of them with the same interpretation as required for the new case was sufficiently like the new case to warrant broadening one of the rules used for that case. The modified rule was then tested on the cornerstone cases to check that all cases were still interpreted correctly. If the case was not close to one of the other case that satisfied the new rule, further conditions were added to the new rule so that it would not fire on any other cases.

It should be noted that more complex changes than those above may be introduced to make the knowledge base more intelligible, more 'organised' but they are not necessary to deal with the maintenance. More complex changes reduce to a series of the above changes. It is worth noting that a more ordered and intelligible knowledge base has nothing to do with facilitating maintenance. A very flat knowledge base with few or no intermediate assertions made and the two maintenance steps outlined above is sufficient for maintenance.

The knowledge dictionary

The above comments on what is necessary to maintain a knowledge base are made with the benefit of hindsight, after the development of a number of maintenance tools. The first tool we developed was the knowledge dictionary, an extension of a conventional data dictionary (Dolk 1988) The motivation for this development was three fold (Jansen and Compton 1989; Jansen and Compton 1989). Firstly it was intended to provide conventional software engineering support in terms of names and definitions of entities and their relationships. The dictionary essentially allows a normalised storage of entities and relationships in that any entity and its relationships can be stored in one place. However it depends on the skill of the data base designer, as to how well this is achieved. The model for the dictionary we developed (Jansen and Compton 1989; Jansen and Compton 1989) is shown in fig 1.

Secondly, and most importantly the dictionary was meant to provide excellent browsing facilities to facilitate knowledge maintenance. To this end instead of a rule being a composite entity composed of conditions and a conclusion, rules in the dictionary were broken into their component parts (fig 2). This meant that very flexible browsing was possible. The user can ask which rules use certain facts but not other facts and reach certain conclusions etc. (Jansen and Compton 1989; Jansen and Compton 1989) These features are not dependant on a special rule parser, they are simply dependent on pattern matching and the use of tables. Other facilities such as *why not* which can be used in conjunction with inferencing to explain why a rule was not satisfied depend on pattern matching rather than special tools to support the expert system. Thirdly the dictionary was meant to simplify the integration of knowledge bases and data bases.

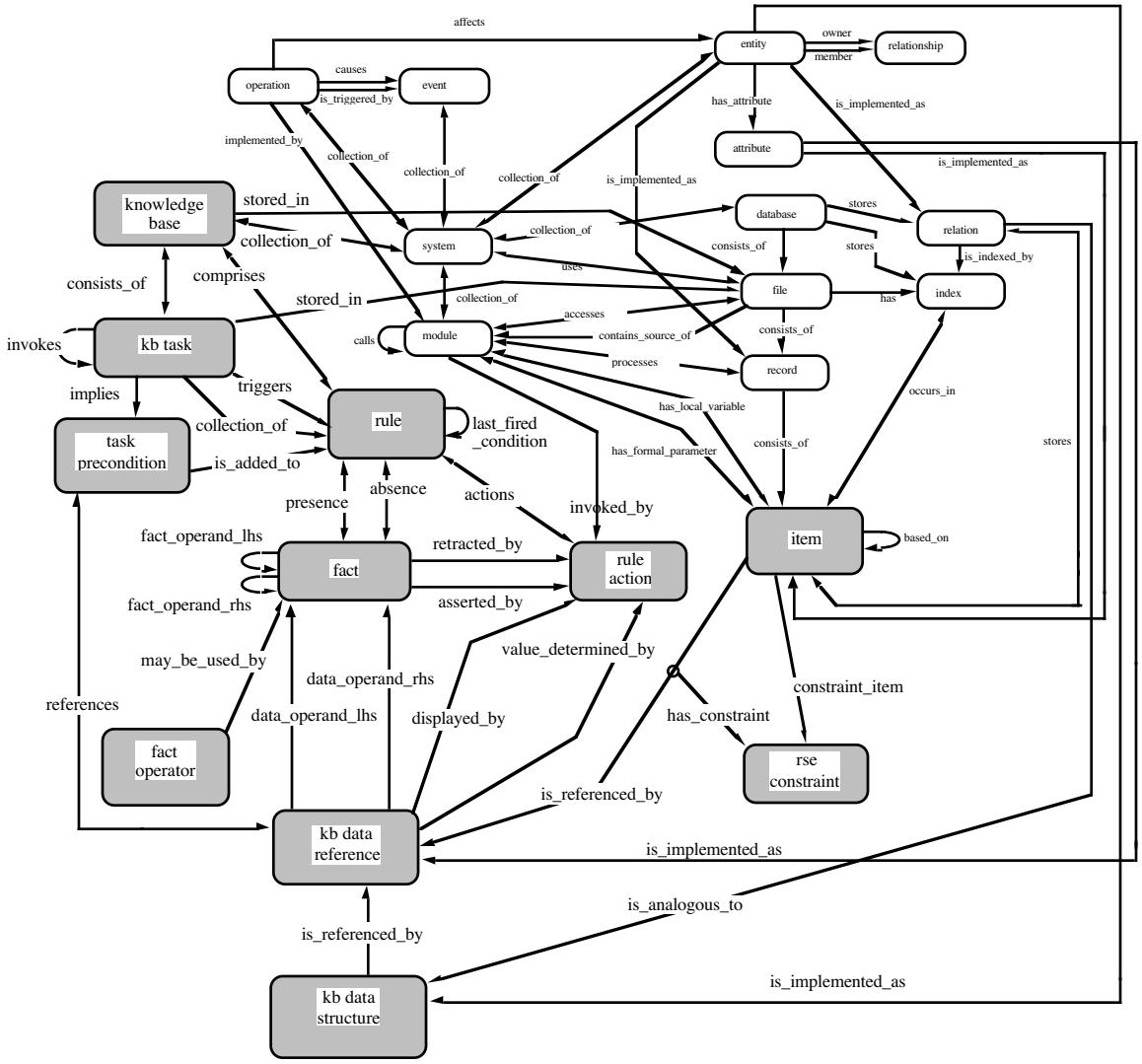

Fig 1 illustrates the extensions to a conventional data dictionary to cover a knowledge base. The extensions are shaded and larger.

Various versions of the dictionary were developed, initially in Prolog, later in Hypercard and finally in Oracle with a Hypercard interface (Lee 1990). The initial design aim of the dictionary would that it would be used for maintenance but not for routine use. Rather a run time version would be generated from the dictionary. Apart from other considerations inferencing in the dictionary is too slow for production use. To speed up inferencing in the dictionary, rules are placed into different categories, depending upon the number of levels of intermediate conclusions that rules produce. Ordering the rules in this manner effectively creates an index to simplify rule analysis and increases the inference engine speed (Lee 1990) and results in an inference speed of less than one minute per case with a 650 rule knowledge base and data with up to nine attributes. The initial development of the dictionary was directed towards rule based system, but ways of handing other representations such as frames and semantic nets were also developed (Jansen and Compton 1990)

PRODUCTION RULE

RULE(42)

IF FTI is high
and T3 is high
and TSH is undetectable
and not on_t4

THEN **thyrotoxic**

ELEMENT RELATIONSHIP TABLE

owner	relationship	member
-------	-------	-------
-------	-------	-------
RULE_42	presence	FTI_high
RULE_42	presence	T3_high
RULE_42	presence	TSH_undetect
RULE_42	absence	on_t4
RULE_42	outcome	thyrotoxic
-------	-------	-------
-------	-------	-------

Fig 2 illustrates a conventional rule representation and the same rule broken into its component parts and represented in a table.

Dictionary limitations

Although the dictionary solves many of the maintenance problems it is not a complete solution. It solves the same sorts of maintenance problems as dictionaries for conventional systems and provides very powerful knowledge browsing facilities, but it is not a sufficient resource for the knowledge engineer and expert to ensure that appropriate changes are made to the knowledge base. The problem is that experts always provide their knowledge in a specific context and the knowledge can only be relied upon in this context (Compton and Jansen 1990).

In observing medical experts and the way they provide knowledge, we came to the conclusion that the knowledge experts provided was not inadequate because they were not good at reporting on their deep mental processes. On the contrary, it was clear that the experts did not attempt to report on their mental processes; they never stopped to reflect "now why do I think that". Rather, they gave a response aimed at satisfying the concerns of the inquirer or knowledge engineer. If they did stop to reflect this was rather to ensure that their reply made sense, that it did not have major gaps or mistakes in it. What they are doing is providing a justification for their judgement, but a judgement tailored to the context, for example the context of whether a peer, a patient or a knowledge engineer is querying their judgement. We have also argued that this justification is essentially "made up" for the occasion, but it is not made up arbitrarily because the expert recognises and checks that it makes sense; that is he or she exercises "insight" (Compton and Jansen 1990).

This model of knowledge suggests that knowledge engineering problems will arise when knowledge is taken out of context. However context is not necessarily specified in the dictionary. An important part of the context is the specific case which has been misinterpreted by the expert system and for which the expert is making up a new rules. For example if the case is **A,B,F,G** the expert might make up a rule

IF **A** and **B**
Then conclusion 5

There may also be a rule in the knowledge base

IF **C** and **D**
Then conclusion 4

This rule will not be found as a possible problem using the dictionary because it bears no relationship to the current case or the new rule added. Also, because an expert's rules tend to be a justification of his conclusions in a specific context, the expert will probably fail to recall that **A,B,C,D** is a common data profile which was correctly classified as conclusion 4 but is now classified as both conclusion 4 and conclusion 5. If the expert is very wise he or she may recall that **A,B,C,D** will be miss-classified by the new rule, but there is no method for the expert to recall this, it is purely luck. Browsing cannot reveal this sort of information, because there is no mechanism to realise what one must browse for. Automatic testing for rule subsumption, conflict and completeness (Suwa, Scott et al. 1984) are not

appropriate to find this sort of problem. One must consider all possible data configurations and in the 650 rule GARVAN-ES1 system this comes to some five million configurations. Even if a range of heuristics could be used to limit the profiles that had to be considered, one has to ask the expert about speculative situations which may or may not occur in reality. In our experience experts are not good at assessing artificial cases - they are likely to create artificial rules and interpretations, which at very least obscure the genuine knowledge in the knowledge base. The better approach to this problem, available to all expert systems, is to have a data base of test cases to run against the changed rules as described above. The dictionary still has this requirement to test the rules against cases. In evaluating the dictionary by making changes to the GARVAN-ES1 knowledge base, it was always necessary to test the cornerstone case. When a condition was removed from a rule or added to a rule it was impossible to anticipate the actions of this rule in terms of the 300 odd cornerstone cases. It was always necessary to run the cases to find if the interpretation of any cases had been affected by the rule changes. Sometimes no cases were affected, but this was impossible to predict. Experts are unable to make these predictions because the changes being made to the rules more for knowledge engineering reasons than for expertise reasons. These changes are generally of no interest to the expert, they are merely a knowledge engineering requirement of the representation and inference strategy.

In other related work Debenham has proposed a method of normalising knowledge to facilitate knowledge acquisition (Debenham 1989). This work also attends only to rules and thus does not appear to deal with the problem outlined above which can only be identified in terms of case data.

Parallel to the development of the knowledge dictionary we developed another knowledge acquisition strategy explicitly aimed at dealing with the problem of knowledge in context (Compton and Preston 1990; Compton and Jansen 1990; Compton, Srinivasan et al. 1991; Srinivasan, Compton et al. 1991). This method, "ripple down rules" allows new knowledge added to the knowledge base to be used only in the context in which it is added. This development does not replace the dictionary and in fact the new rule structure can be represented in the dictionary (Jansen and Compton 1989) and can be seen in Fig 1 in the *last_fired* relationship between rules. It was found that capturing and using knowledge in context greatly simplified knowledge acquisition and allowed rules to be added 40 times faster than with conventional acquisition. (Compton and Jansen 1990) However even this did not remove the problem completely. The expert was still likely to make up a rule for a misinterpreted case which would also interpret other cases correctly interpreted by an earlier rule on the particular path through the knowledge base leading to the new rule. The only cornerstone case which could be misinterpreted is the case associated with the last rule that fired in the path through the knowledge base, so only one cornerstone case had to be tested. It should be noted that ripple down rules are never altered and a new rule is added for each case that fails, so that there is a cornerstone case directly related to each rule. This is not the case with a conventional system. Rule base changes are made to accommodate particular cases, but the relationship between

individual cases and rules is lost as the knowledge base is developed making it necessary to test all the cornerstone cases.

Since only a single cornerstone case needs to be tested, testing can be bypassed altogether by presenting the expert with a list of the differences between the new case for which a rule is being composed and the case associated with the last rule that fired, the rule that gave the wrong interpretation for the new case. The expert then picks one or more of these conditions, for the rule. Since the conditions represent the differences between the cases this new rule is guaranteed to correctly interpret the new case but not misinterpret cases like the cornerstone case of the last fired rule. This greatly simplifies knowledge acquisition and the expert needs only exercise his expert judgement selecting form the list of conditions rather than trying to guess knowledge engineering requirements. A medical expert system fulfilling these criteria has recently been introduced into routine use (Compton, Srinivasan et al. 1991) in the Department of Chemical Pathology St.Vincent's Hospital for the interpretation of pathology reports. At the time of writing this system has been in routine use for the six weeks and so far 498 rules have been added. Development commenced with thyroid data then moved to catecholamines and is now concerned with acid/base balance. The thyroid development was completely separate from the previous Garvan system an further evaluation of the method. At 230 rules the system is about 95% correct which is consistent with earlier Garvan results and suggest that the system will eventually double in size. However the rate of knowledge acquisition has slowed down to a rule every few days, so that it is clearly in the maintenance phase. There are only a few catecholamine rules to date. The rate at which acid/base rules are added also appears to be starting to slow down.

Although ripple down rules are a powerful knowledge acquisition strategy, they are probably not general enough for all knowledge based systems. For example, there are tasks which require the knowledge base to be ordered into as compact and meaningful a representation as possible, with for example many levels of intermediate conclusions. Ripple down rules allows no reorganisation and no intermediate conclusion. It is strictly aimed at situations where an expert system is required but there is no interest in achieving a particular structure in the rules base. In themselves also ripple down rules provide no browsing, but browsing can be achieved by representing the ripple down rules in the dictionary.

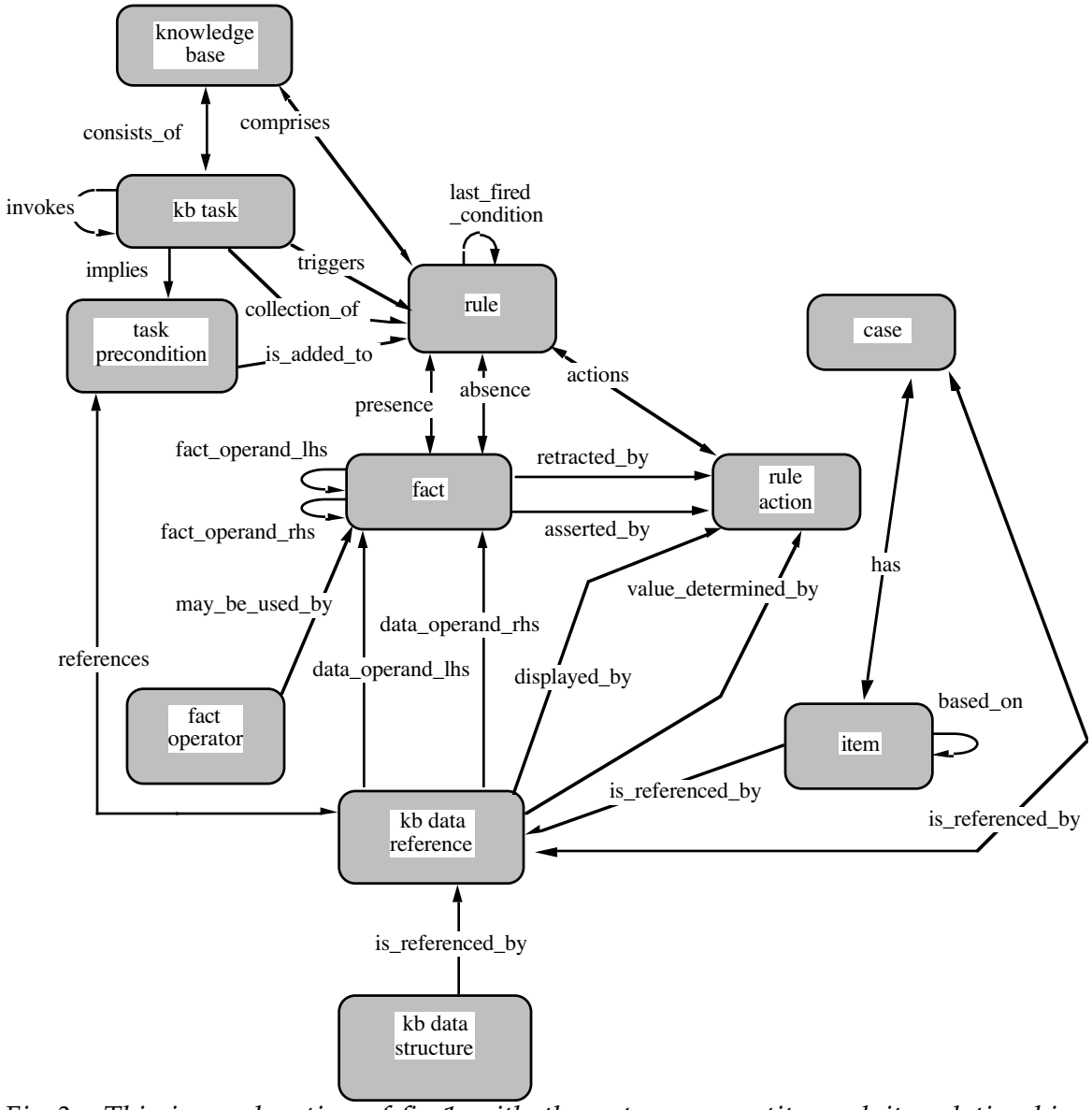

Fig 3. This is a subsection of fig 1 with the extra case entity and its relationships included.

Knowledge dictionary with cases

The obvious solution to the limitations of the dictionary is to include cornerstone cases in the dictionary. Since, as fig 1 illustrates, the knowledge part of the dictionary is be closely integrated with the data base part, there is no reason why the cornerstone cases cannot be directly stored in the dictionary. This means that when one is narrowing a rule by adding a further condition one can query whether any case that normally is satisfied by that rule is lacking the condition so that it will be misinterpreted. Similarly if one is broadening a rule one can query whether any cases will satisfy the rule which are normally interpreted by some other rule. The aim of including cases in the dictionary would be to eliminate the idea of running the inference engine. Rather the cases of interest should be found by data base queries similarly to the way rules are browsed. The cases

determine the maintenance to the knowledge base and so are an integral determinant of the knowledge and should be an integral part of the knowledge base.

The obvious way to include the cases in the dictionary is described in Fig 3. The extra relationships are a many to many relationship between the entities case and kb_data_reference and a many to many relationship between the entities case and item. Each case has many data items, and each particular data item may occur in many cases. Each case also has many kb data references, one of which has a property of an interpretation text. Using these relationships one can always find which rules will fire for a certain case and which cases satisfy a particular rule. One can query what rules have a certain fact as a condition and which cases include this fact. This information is gained by data base queries rather than running the inference engine, or rather running the inference engine can be seen as a data base query.

The problem with the model in fig 3 is that there are a large number of relationships to be traversed for any of the above queries relating cases to rules resulting in poor response depending on the resources available. The studies described here were carried out with an implementation of the dictionary in Oracle running on a Apple Macintosh II with a Hypercard interface. With this particular implementation one inference, running the rules against a single case takes about 1 minute. For the present studies we therefore used a less well normalised model but one which allowed very rapid queries. This meant that after any changes one had to check the relationships to see that they were still correct and if necessary change them. A *save facts* function was developed to ensure the knowledge base was not corrupted. This was not desirable but it made it possible to experiment with the dictionary, where we were dealing with 291 cases and 650 rules. The extensions to the dictionary to deal with cases were implemented in Hypercard.

We have previously described a range of functions for browsing the knowledge base (Jansen and Compton 1989). These have been applied to cases as well. For example one can ask for any entity in the dictionary for its properties and relationships to be shown. This allows related cases and rules to be retrieved. If a rule is changed or added the function *matched cases* allows all cases interpreted or misinterpreted by the rule to be retrieved. A special function is required for this because of the non normalised model we used for speed considerations. An advantage of the non normalised model is that it allows for more easy assessment of the effect of changes. For example the *matched cases* function reports all the cases that satisfy a given rule, but because the cases have relationships with rules that used to be satisfied by the cases the changes can be assessed. However when the changes are complete the relationships of the cases have to be brought up to date.

The utility of this approach was examined by adding a further 11 cases to the Garvan cornerstone cases and making the appropriate rule changes. Some of the cases initially had two interpretations, one correct and one incorrect and the incorrect one had to be prevented. In other cases a wrong interpretation was

made, and had to be replaced by the correct interpretation. As expected the extension of the dictionary to cover case facilitated this knowledge maintenance.

Discussion

With hindsight it seems very obvious that the cases that require changes to the rules play an integral part in determining these changes. It is then also obvious that these cases should play an integral part in determining any further changes to the same rules for which they were introduced. Rather than just running the cases as test cases it seems reasonable that they be stored in the dictionary as an integral part of the knowledge base.

The question of how the cases should be integrated into the dictionary is still open and the approach we have outlined in fig 3 is tentative at best. It seems that the best way of including the cases would be to use them to determine what kinds of changes the knowledge engineer or expert may make to the knowledge base. We have previously noted the ripple down rule methodology (Compton, Srinivasan et al. 1991) where the conditions that can be included in the new rule are determined by the differences between the case for which the rule is being added and the case for which the last rule fired was added. For example we may have a rule which has an associated case, both identified as 45:

RULE 45
IF **A** true and
B true and
THEN . . . . . .

When a new case is misinterpreted by RULE 45 it is compared to Case 45 and the expert chooses conditions from the differences between the cases.

Case 45	New case
A true	**A** true
B true	**B** true
D true	
E true	**E** true
	F true
	G true

A new rule which is guaranteed to correctly interpret the new case, but not misinterpret the old case could be composed from any combination of the difference between the two cases. A rule with all the differences would be:

IF **D** false and
F true and
G true
THEN . . . . . .

With this approach the expert pays no attention to knowledge engineering. He or she just picks the most suitable conditions from a list of differences. As long as one difference is picked, other conditions which are true for both cases can be added for clarity, if desired.

There is no reason why this approach has to be restricted to the ripple down rule structure. Initially if we assume that there are no intermediate conclusions, then we have the simple situation outlined in the introduction of any maintenance incident requiring the narrowing of a rule and/or the broadening or introduction of another rule. If the rule to be narrowed has a case associated with it, then the condition to be added to the rule must be a condition that exists in the old case but not in the new. As the knowledge base expands there will be a number of cases associated with each rule. In narrowing a rule by adding conditions the condition must come from the intersection of all the cases associated with the rule but not from the new case. In broadening a rule, the conditions to be removed must be one or more of those in the rule that do not occur in the new case while the remaining conditions in the rule must not occur in any other cases in the data base. Lists of these possible conditions should be presented to the expert for him or her to choose and the new rule automatically added to the knowledge base and new relationships automatically added. Fig 4 shows the modified conceptual model to deal with this type of automatic knowledge maintenance.

This approach to maintenance is explicitly dependent on cases. This is not a problem as all maintenance occurs because of misinterpreted cases. There is no such thing as maintenance for no reason. In the early stages of maintenance before there are cases associated with a given rule the difference between the new and the old cases would of course include all the conditions for the new case. As the knowledge base developed and cases became associated with rules the difference lists would become more specific.

A further question arises as to identifying which rule is a suitable candidate for being broadened to interpret the new case. Once the interpretation for the case is identified the system could find all rules that gave this interpretation and identify which conditions in these rules did not occur in the case and of the remaining conditions in the rules which did not occur in any other cases in the data base. The various groups of conditions which might be selected (normally only one) would then be displayed and choosing one of them would automatically determine which rule was to be broadened. If no rules could be found the expert would be prompted to make a new rule. The conditions displayed for this rule would be the difference between the present case and all other cases in the knowledge base. To simplify the list of *not* conditions, as conditions were added to the rule the possible cases which could be used to generate the difference list would be reduced, as fewer and fewer cases satisfied the developing rule, resulting in a smaller list as each condition is added.

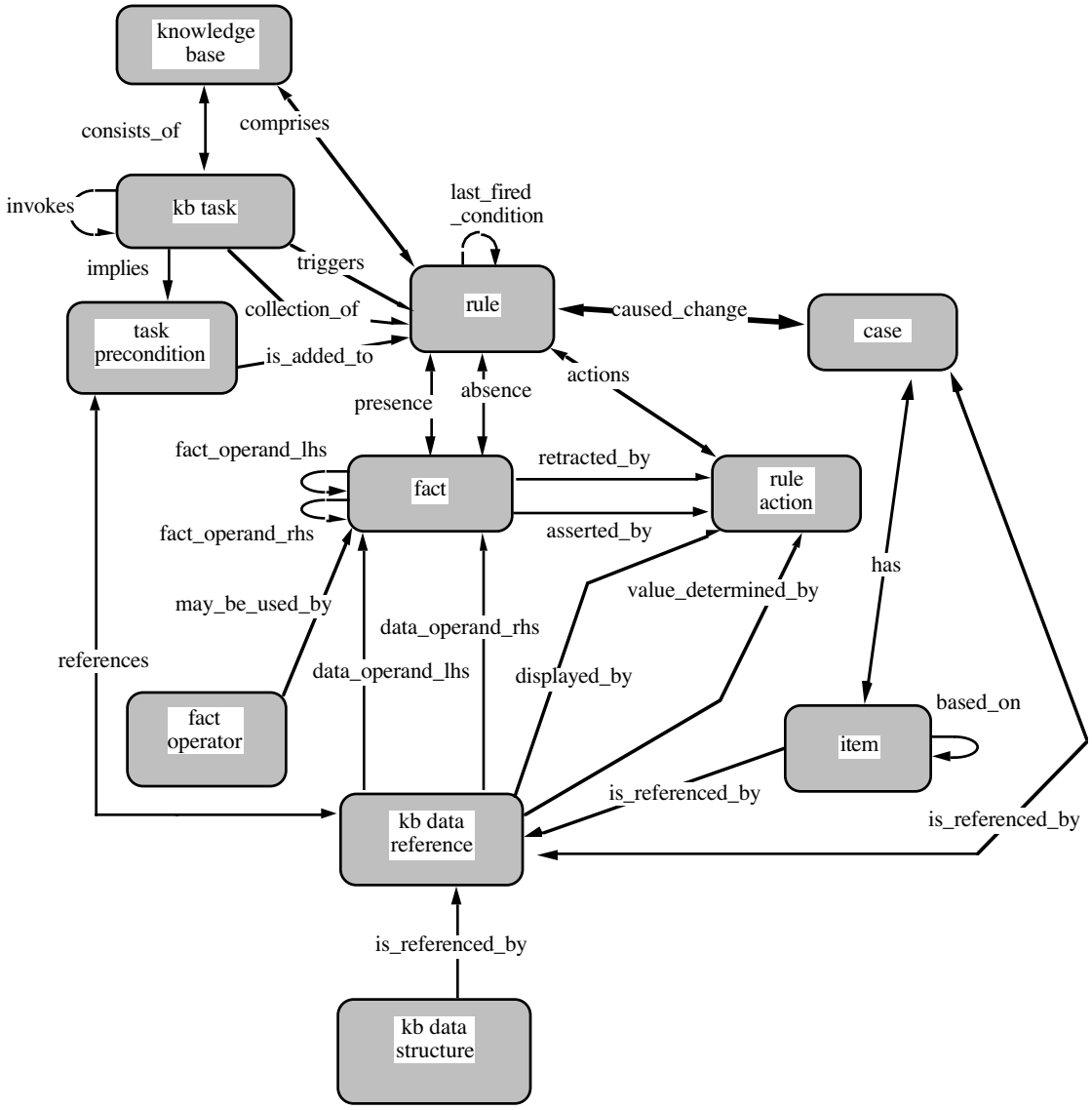

Fig 4. This illustrates a further extension to the conceptual model to include cases as in fig 3 and the "caused change" relationship between cases and rules which can be used to retrieve the appropriate cornerstone case when further changes are being made to a rule.

We have described here an automated approach to maintenance which assumes a flat expert system. We hypothesise that the same approach can probably be applied to any rule based system. When a case is run on the expert system, a rule trace can be provided of the sequence of rules that have been traversed. Each of these rules will have cases associated with them. We assume that with a more complex knowledge base a case will have a relationship not just with the rule that was changed to accommodate it, but with all rules that fire when this case is run. The same type of strategy as above can then be applied to deciding what conditions should be added to or removed from a given rule. The important extension to the approach for a more complex system is that a number of rules may be able to be changed to bring about the same change in performance of the expert system. The decision of which rule to change could be partly handled by the expert's choice of conditions from the condition list as

before, but metarules could also be used. These rules would be chosen to produce a desired structure in the rule base.

Broadening rules is more difficult. One can again find all the rules that produce the desired conclusion and then retrieve all the cases associated with these rules and thus all the other rules that led to these conclusions. One could then find which conditions from these rules do not occur in the case for which the changes are being made, such that the remaining conditions in the rule do not occur in any other cornerstone case. (Note, as in previous dictionary work, we assume rules should only have conjunctions of conditions not disjunctions.) Heuristics could again be used to decide which rule should be changed if there is more than one candidate. For example, if one wished to be as conservative as possible, rules at the end of an inference chain would be broadened, whereas if one was less concerned with errors and wanted rapid coverage of the domain, early rules in an inference chain would be broadened.

It does not seem necessary to take particular account of the conflict resolution or the inference strategy in this process. The key feature of the process is that the behaviour of the system should be able to be identified from looking at the cornerstone cases it deals with. The maintenance changes attempt to preserve the classifications of these cases giving incremental and controlled changes to the knowledge base. Strictly, the conflict resolution strategy or probabilistic reasoning method are not controlled for and may cause unexpected changes in the performance of the knowledge base. However, if this is the case we would contend that the knowledge base is probably unmaintainable. Certainly conventional approaches to changing the knowledge would be even more susceptible to problems than the method we have outlined. It is also worth raising the question of whether the availability of this sort of approach to maintenance would change the demand for sophisticated inferencing and conflict resolution strategies. Ultimately these strategies are only desirable in the hope that they will simplify the knowledge acquisition and engineering in that the system uses the inference strategy rather than specific knowledge.

The method we have proposed is starting to appear to be on the same scale of endeavour as methods of exploring all the logical possibilities of the knowledge base. There is an important difference between the two, in that here we are using a set of specific cases to control the knowledge acquisition, whereas logical methods tend to explore all possible data configurations. We assume that the case base method will be more manageable.

What we have suggested here is based on the same principle underlying ripple down rules that knowledge acquisition should always be in context. With ripple down rules the context is determined by the pathway through the knowledge base and the difference between the case associated with the last fired rule and the new case. What is proposed here is a more general solution but where the possible rule changes are again determined by the difference between the new case and previous cases associated with rules along the inference pathway. With ripple down rules, the only difference that mattered was between the new case and the case associated with the last rule in the chain because the new rule could

only be added at the end of the chain. Here the location of the change in the rules would be determined by heuristics chosen to produce a certain type of knowledge structure. Ripple down rules are very easy because they depend on a single difference, but they are likely to produce redundant rules. They are a limiting case of the more general knowledge in context method we are proposing here. At this stage we have no idea of the relative costs of the different approaches. This new proposal would may allow redundancies to be reduced and a variety of structures to be produced, but at the cost of a too sophisticated maintenance process. Ripple down rules are very simple and fast and in the test domain of pathology redundancy is not yet a significant problem.

It is important to note that this approach is not related to machine learning. The expert still makes the decisions about what knowledge should be used in rules. However, his or her choices are determined by what conditions are possible in the context to produce a suitable change in the knowledge base. The expert is thus required to make decisions relating to expertise rather than knowledge engineering. There is no knowledge engineering involved except in deciding on the heuristics which will determine the overall structure of the knowledge base. Normal methods of building expert systems are weak in that the interaction between the expert and knowledge engineer determines the structure of the knowledge base. This structure is implicit rather than explicit and may change with circumstances, changes in personnel etc. With ripple down rules and the approach outlined above there is a clear separation between engineering and expertise. The structure of the knowledge is fixed and is determined by the heuristics put in place to identify which rules should be modified when a choice is possible. Underlying this approach is the idea that a knowledge base is a model of reality rather than a representation of expert knowledge (Clancey 1989). Any good model needs a fairly constrained methodology for how the model is to be built. The expert because he or she is an expert is able to answer questions of expertise within the constraints of a variety of modelling methods. The underlying strategy of asking the expert about case differences has been argued here in terms of context. It is closely related to the hypothesis that human experts are best at identifying differences between concrete object which underlies KSSO and other knowledge acquisition tools based on Kelly's personal construct theory (Gaines and Shaw 1990)

Summary

We proposed that case data is integral to determining the development of a knowledge base and so should be included in the knowledge base to facilitate maintenance. We have developed a simple, but limited way of achieving this integration within the knowledge dictionary. We have then proposed that if such an integration is fully developed it would allow for greatly simplified knowledge maintenance. The expert would still be responsible for creating and modifying rules, but his task would be reduced to selecting from a list of conditions guaranteed to produce a correct rule from the engineering perspective.

References

Buchanan, B., Barstow, D., Bechtel, R., Clancey, W., Kulikowski, C., Mitchell, T. and Waterman, D. (1983). Constructing an expert system. Building expert systems. Reading Massachusetts, Addison Wesley. 127-67.

Clancey, W. (1989). "Viewing knowledge bases as qualitative models." IEEE Expert Summer: 9-23.

Compton, P., Horn, R., Quinlan, R. and Lazarus, L. (1989). Maintaining an expert system. Applications of Expert Systems. London, Addison Wesley. 366-385.

Compton, P. and Preston, P. (1990). A minimal context based knowledge acquisition system. "Knowledge Acquisition: Practical Tools and Techniques" AAAI-90 Workshop, Boston,

Compton, P., Srinivasan, A., Edwards, G., Malor, R. and Lazarus, L. (1991). Knowledge base maintenance without a knowledge engineer. Proceedings of the 1st world congress on expert systems, Orlando, Pergamon.

Compton, P. and Jansen, R. (1990) "Knowledge in context: A strategy for expert system maintenance." Proc AI 88. Ed. CJ Barter and MJ Brooks. Lecture notes in artificial intelligence 406. Berlin: Springer-Verlag, . 292-306.

Compton, P. J. and Jansen, R. (1990). "A philosophical basis for knowledge acquisition." Knowledge Acquisition 2: 241-257.

Debenham, J. (1989). Knowledge systems design. Prentice Hall.

Dolk, D. (1988). "Model management and structured modelling: the role of an information resource dictionary system." Communications of the ACM 32(6):

Gaines, B. and Shaw, M. (1990). Cognitive and Logical Foundations of Knowledge Acquisition. AAAI Knowledge Acquisitioh Workshop, Bannf,

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

Jansen, R. and Compton, P. (1989). "The knowledge dictionary: a data dictionary approach to the maintenance of expert system." Knowledge Based Systems 2(1): 14-26

Jansen, R. and Compton, P. (1989). "The knowledge dictionary: storing different knowledge representations." Proc 5th Aust Conference on Applications of Expert Systems : 143-162.

Lee, M. (1990). The implementation of a knowledge dictionary in SQL. Oracle Asia/Pacific User Conference.

Srinivasan, A., Compton, P., Malor, R., Edwards, G. and Lazarus, L. (1991). Knowledge Acquisition in Context for a Complex Domain. Proceedings of the Fifth European Knowledge Knowledge Aquisition Workshop. Pergamon. in press.

Suwa, M., Scott, A. and Shortliffe, E. (1984). Completeness and consistency in a rule-based system. Rule-based expert systems. Reading Mass, Addison Wesley. 159-70.

van de Brug, A., Bachant, J. and McDermott, J. (1986). "The taming of R1." IEEE Expert Fall: 34-38.