PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

THE KNOWLEDGE DICTIONARY: A RELATIONAL TOOL FOR THE MAINTENANCE OF EXPERT SYSTEMS

Bob Jansen* **& Paul Compton†**

* CSIRO,Division of Information Technology,PO Box 1599,North Ryde,NSW 2113,Australia.
ACSnet: jansen@ditsyda.oz JANET: ditsyda.oz!jansen@ukc
ARPA: jansen% ditsyda.oz@ seismo.css.gov CSNET: jansen@ditsyda.oz
UUCP: {enea, hplabs, mcvax, prlb2, seismo, ubcvision, ukc}! munnari !ditsyda.oz!jansen
AUSPAC: jansen@au.csiro.ditsyda
† The Garvan Institute of Medical Research,St. Vincents Hospital,Darlinghurst,NSW 2010,Australia

ABSTRACT

In this paper, we discuss the development and use of a *Knowledge Dictionary*, a tool to facilitate the documentation and maintenance of rule based expert systems. The *Knowledge Dictionary* may be used to record heuristics and their component parts, facts and rule actions,utilizing the relational data model to store the heuristics in a data form rather than executable code form. Thus knowledge based systems can be documented in the same way as conventional systems. Use of the relational model allows examination of the knowledge in a knowledge base to be completely flexible, in contrast to conventional expert system tools. This flexibility is essential if the Feigenbaum bottle-neck is to be relaxed and if maintenance is to proceed in an orderly and logically consistent fashion. This paper discusses the use of the *Knowledge Dictionary* to redevelop GARVAN-ES1, a medical expert system for the automatic clinical interpretation of laboratory reports.

1. INTRODUCTION

Long term use and maintenance of knowledge based systems is still a largely unknown area. There is almost no documented study of the maintenance problems with the exception of XCON/R1 (Bachant&McDermott 84) and GARVAN-ES1, (Compton *et al* 88). However there is no doubt that maintenance problems will be of major importance to the large numbers of expert systems now being deployed.

XCON provides data on the maintenance required by an expert system whose range of expertise is continually expanding due to the introduction of new computer systems it is required to configure (Bachant&McDermott 84). In contrast, GARVAN-ES1 has had no change in its range of expertise only an increase in its level of expertise.

GARVAN-ES1 is a medical expert system which provides automated clinical interpretation of diagnostic reports from a pathology laboratory (Horn *et al* 85, Compton 87). The current system is restricted to thyroid interpretation. and has been in production since 1984. When introduced 96% of its interpretations were acceptable to domain experts. Currently 99.7% of its interpretations are accepted and the rule base has doubled in size. Figure 1.1 provides an example of the growth of a single rule over this period and provides a good illustration of the maintenance problem, when a system is only being refined, not expanded.

Formal maintenance strategies are required to cope with maintenance on the scale that this example suggests. We propose that many of the conventional software engineering tools, such as formal specification, modelling, and data dictionaries, used daily in the engineering of conventional database systems are applicable to AI system also.(Jansen 87, see also Debenham 85, Debenham 86 & Duda *et al* 87). In particular we propose to apply the dictionary concept to knowledge as well as to the information/data areas. This requires the rules to be stored as data rather than as directly executable code.

This proposal differs from other related work using the dictionary concept to couple expert and database systems (eg. Al-Zobaidie *et al* 87, Dolk *et al* 87, Leung & Nijssen 87, Ishikawa *et al* 86, Held & Carlis 85), in that we propose that the one integrated dictionary should encompass all aspects of a system, both knowledge base and data base. Stand alone systems such as NEXPERT OBJECT have interfaces to relational data bases but thereby require a duplication of definitions.

In our work towards a *Knowledge Dictionary* to act as the central pivot for integrated system design we have so far completed the design and implementation of a dictionary augmented to cater for the modelling of production rules using the relational data model formalism, and which can generate Prolog data structures enabling inferencing on those data structures. This paper discusses the design of the *Knowledge Dictionary*, some features of its implementation in Prolog and future directions of this research.

1984

```
RULE(218)
IF    ethy_utsh and TT4_BORD isnt high
      and no_comment
THEN DIAGNOSIS ("... thyrotoxicosis & non-thyroidal illness")
```

1988

```
RULE(21800)
IF    ethy_utsh and (TSH is undetect or TSH is low)
      and ((FT4_BORD is missing and FTI_BORD is missing)
      or (TT4 is high and TT4_BORD is missing))
      and (no_comment
      or ((sick or hyperthyroid)
      and not (on_t4 or query_t4 or surgery or tumour)))
      and not screening
THEN DIAGNOSIS("... thyrotoxicosis & non-thyroidal illness")

RULE(21801)
IF    ethy_utsh and ((TSH is undetect or TSH is low)
                    or TSH is missing)
      and (FT4_BORD is high or FTI_BORD is high)
      and (TT4_BORD is high or TT4 is missing)
      and T3_BORD isnt high
      and (no_comment
      or (sick and not (on_t4 or query_t4 or surgery or tumour)))
THEN say_discordant  NOW TRUE
```

Figure 1.1 - An example of rule expansion due to refinement of the knowledge over time

2. THE PROPOSED MODEL

The *Knowledge Dictionary* is based on the *Entity-Relationship* (ER) model as shown in figure 2.1[1].

The model details the object types recognized by the dictionary, and the relationships between the object types. This model is user extensible, and thus the user can add any *object types/allowed relationships* to this model to cater for other requirements (causal models, fuzzy logic, temporal reasoning etc.).

The majority of this model is standard for a conventional data dictionary[2]. The extensions we have made are highlighted as the shaded entities. It should be noted that the diagram shows what relationships an object type *may* make with its surroundings, called *allowed relationships*. The actual relationships for any object occurrence depend on its usage. In addition, the nature of the relationship is not shown on the diagram. A relationship can be pointer, or set based, as in a Codasyl database, relational, or value based as in the relational data model, or even function based, where the membership of a relationship is dependent on the evaluation of some function, returning a true or false condition as appropriate. In each case, the relationship has properties defining the relationship type. For set based relationships, properties include sort sequence and keys for sorted sets, set order, connect and disconnect requirements as mandatory or optional, etc.

[1]Note that this model is the conceptual model. The actual implementation is a meta model of this model. This allows the conceptual model to be extensible without altering the dictionary functions themselves.

[2]The model diagram as shown is certainly not complete for the conventional side. This is addressed by other software engineering research within the CSIRO Division of Information Technology. Dolk *et al* 87 describes the work being undertaken by ANSI to produce a standard model for a resource dictionary system, and this model corresponds closely with our model for the conventional side.

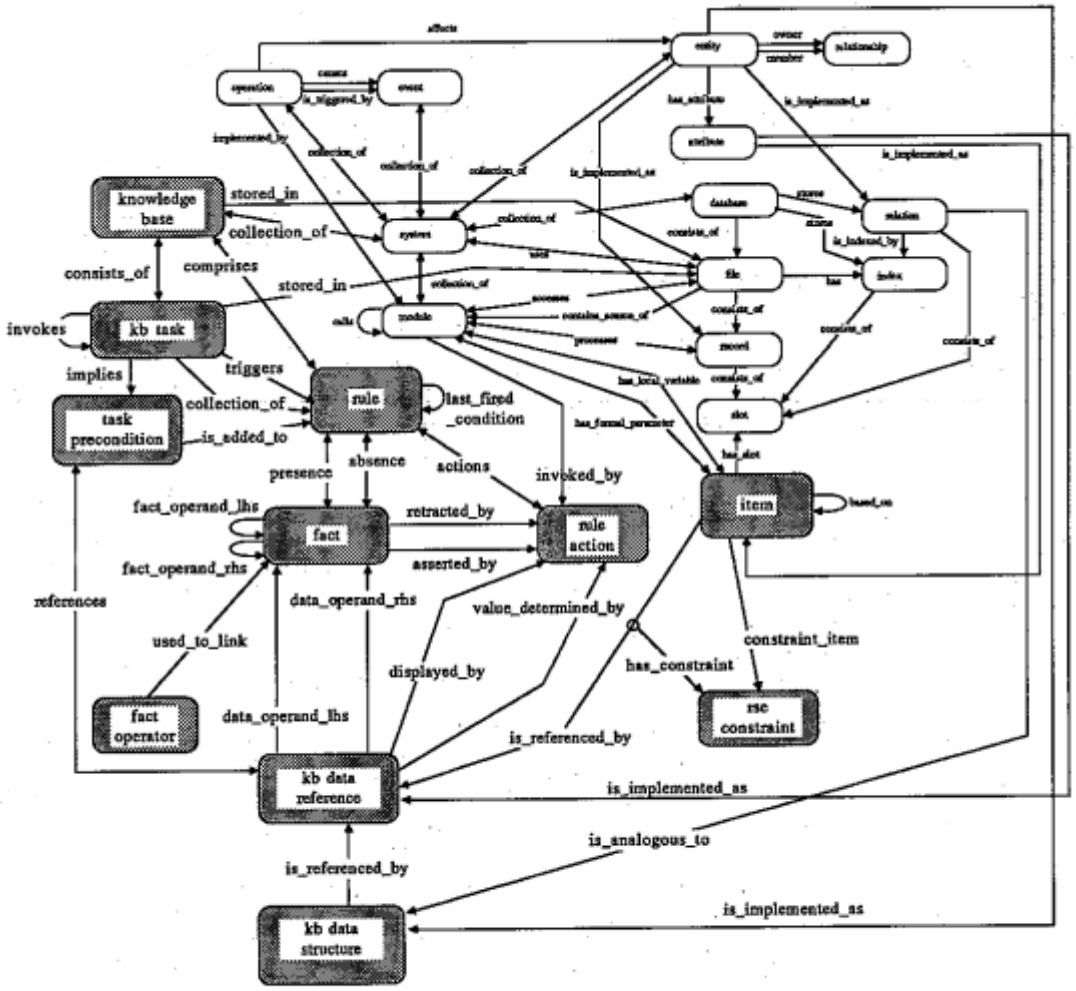

Figure 2.1 - Knowledge Dictionary Conceptual Diagram

The idea underlying the model is to treat each object type as a set of data or a table, thus allowing standard data manipulation operations upon it. For example, as the *Knowledge Dictionary* is currently implemented in Prolog (see below) using the *Relational Data Model* (Codd 70), all the standard relational operators (union, intersect, difference, select, project, join, divide, and HAS (Carlis 86)) are potentially available to the knowledge engineer to browse and maintain the knowledge. The use of the relational operators on the data representation of the rules, allows a rich browsing and exploration capability. This is not normally available for an expert system.

The model recognizes that rules themselves have a structure. As shown in figure 2.1, rules test for the presence or absence of a set of facts, and if the fact profile is matched, then a set of rule actions is performed. Each rule action may assert a fact, retract a fact, display some data item, call a code module, or access a data record in some data store etc. It should be noted that our model enforces disjoint (or disjunctive) normal form on rule structure. This ensures that each rule is simpler to understand, in addition to excluding the problems associated with mixing NOT and OR conditions. However the underlying dictionary allows groups of rules with the same action to be examined in concert.

An important aspect of the system is the fact structure. The use of intermediate facts can be a requirement of the implementation, sometimes to the point where the expert can no longer recognize the knowledge they have supplied. But they can also be necessary to allow the expert's knowledge to be captured as expressed, an essential requirement if the expert's familiarity with their knowledge is to be retained. An expert will lump together groups of facts in a single profile or a more complex fact, which may be viewed in terms of Clancey's concept of abstraction (Clancey 85).

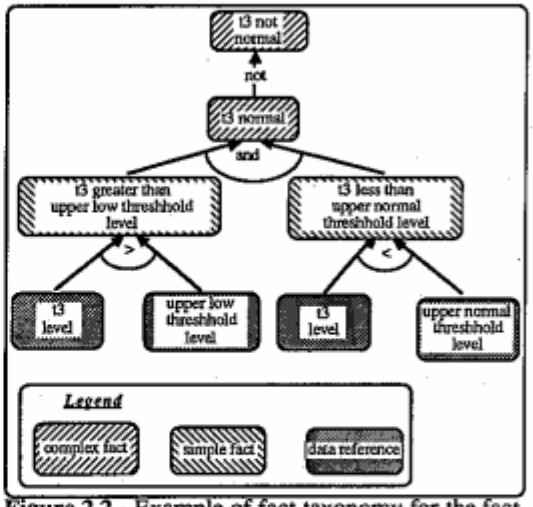

Figure 2.2 - Example of fact taxonomy for the fact *t3_not_normal*

We consider facts as belonging to a *fact taxonomy*[1], and this taxonomic structure can be used to record under which conditions a particular fact is true[2]. Figure 2.2 shows an example of this taxonomy for the fact *t3 not normal*. Classification terms include *simple* or *complex*, and *internal* or *external*. *Complex* facts consist of a fact list where member facts are related by the operators AND or AND NOT. *Simple* facts are those facts that are true depending on an algorithm involving data references, not other facts. *Internal* facts, having no associated concept in the knowledge domain, are those facts defined by the knowledge engineer to aid in the knowledge implementation and representation, whilst *external* facts are those facts that have associated concepts in the knowledge domain. Thus, in figure 2.2, the facts *t3 not normal* and *t3 normal* are complex external facts, whilst the facts *t3 greater than upper lower threshold level* and *t3 less than upper normal threshold level* are simple internal facts

[1]The fact taxonomy can be can be represented by a connected graph structure.

[2]Some facts become true when they are asserted by the action of a rule, modeled in the Knowledge Dictionary by forming an *asserted_by* relationship with the appropriate *rule action* object. In this case, they are included in the taxonomy only if they are a pre-requisite for the assertion of another fact.

In the *Knowledge Dictionary*, this taxonomy is stored as follows. The fact object is given a property of *operator*, which stores the operator used to relate the member facts or data items. Complex facts are related to their 'simpler' facts by the *fact_operand_lhs* and *fact_operand_rhs* relationships. Simple facts are related to their data items via the *data_operand_lhs* and *data_operand_rhs* relationships (See figure 2.1).

Thus by starting at the top level fact, and storing the operator and the appropriate relationships, the complete taxonomy can be stored one level at a time. When requested, the structure can be evaluated or displayed by traversing the relationship linkages starting from any specified fact and applying or displaying the stored operator. In the case of simple facts, the appropriate data item may have to be retrieved from the data store, and thus standard gateways should be invokeable automatically[1] (Jansen 88).

3. IMPLEMENTATION DETAILS

The prototype *Knowledge Dictionary* has been built in AAIS Prolog running on a Macintosh II computer. Currently, the prototype *Knowledge Dictionary* contains approximately 624 knowledge domain rule objects, 217 fact objects, 185 rule action objects, and 3800 relationship tuples, constituting the re-implemented version of the GARVAN-ES1 thyroid interpretation expert system in disjoint (or disjunctive) normal form. The following details how the rules are stored in the *Knowledge Dictionary*.

Take as an example the rule from the existing GARVAN-ES1 thyroid expert system as shown in figure 3.1. Prior to inserting this rule in the *Knowledge Dictionary*, the structure of the rule needs to be elicited. In this case, we have the objects as shown in the bottom of figure 3.1.

Initial Rule

RULE(11100.01)
IF T3 is high
and hithy
and TSH is missing
and TBG isnt high
and T4U isnt high
and surgery
and not on_t4

THEN DIAGNOSIS("The T3 and THY are elevated consistent with thyrotoxicosis")

Constituent Objects

a rule object named 11100.01

a number of fact objects, named
t3_high
hithy
tsh_missing
tbg_not_high
t4u_not_high
surgery
on_t4

a rule action object given an internal name of interpretation_1

Figure 3.1 - An example rule, showing its decomposition into constituent objects[2]

Note that the naming convention in the *Knowledge Dictionary* model is based on a unique type/name doublet for any object. Thus, as in this case, objects of different types may have the same name. These objects would be stored as shown in figure 3.2.

With the need to name each object, we import the problems of naming conventions and 'meaningfulness'[3]. From the above, it can be seen that if the names of the objects are meaningful in their own right, descriptions may not be required. However, they may be entered for each object occurrence to improve understanding. These descriptions are presently stored in flat files accessible through the standard AAIS Prolog 'edit window' facility.

Similarly, the names of the *allowed relationships* from figure 2.1 forming the first parameter in the *element_relationship*

[1]The actual form of the interface between the knowledge base and the database is still an area of much research. However, it is desirable that the data retrieval statements used to retrieve the data from the database should be stored in some form in the *Knowledge Dictionary*, so that they are available for the maintenance process.

[2]Note that in this diagram, the fact *surgery* is an internal fact, and refers to the patient having had some form of thyroid surgery.

[3]The Knowledge Dictionary currently does not recognize internal or external representations of names, but this is to be added, prior to the next prototype.

table, are hopefully meaningful, so the purpose of the relationship is clear. Any object occurrence may also have an associated description.

```
element(rule,'11100.01').
element(fact,t3_high ).
element(fact,hithy).
element(fact,tsh_missing).
element(fact,tbg_not_high).
element(fact,t4u_not_high).
element(fact,surgery).
element(fact, on_t4).
element(rule_action,interpretation_1).
element(kb_data_reference,interpretation_1).
element_property(kb_data_reference,interpretation_1,
   "The T3 and THY are elevated consistent with
   thyrotoxicosis").
element_relationship(presence,rule '11100.01',
                     fact,t3_high).
element_relationship(presence,rule,'11100.01',
                     fact,hithy).
element_relationship(presence,rule,'11100.01',
                     fact,tsh_missing).
element_relationship(presence,rule,'11100.01',
                     fact,tbg_not_high).
element_relationship(presence,rule,'11100.01',
                     fact,t4u_not_high).
element_relationship(presence,rule,'11100.01',
                     fact,surgery).
element_relationship(absence,rule,'11100.01',
                     fact, on_t4).
element_relationship(actions,rule,'11100.01',
                     rule_action,interpretation_1).
element_relationship(displayed_by,kb_data_reference,
                     interpretation_1,rule_action,
                     interpretation_1).
```

Figure 3.2 - Example of object and relationship storage in the *Knowledge Dictionary*

4. AVAILABLE FUNCTIONS (SOME)

Using the above relational data declarations (eg. figure 3.2), a number of functions have been made available to manipulate the data[1]. Even at this stage the functions have been provided with a simple translation facility, to ensure that the captured knowledge is accessible to the experts in their familiar formalism; e.g. for entering and displaying the rules, the production rule formalism, IF...THEN..., has been chosen. Currently, this translation between the internal and external formalisms is hard coded into each function but will ultimately be stored as *meta knowledge*, thus allowing a general translation mechanism for different external formalisms.

The USAGE function is used to determine who uses what and how. It searches the *element_relationship* table extracting any tuples satisfying the criteria, and displays the result to the user.

The SHOW_RULE function displays the specified rule on the user's terminal, in the familiar IF...THEN... form.

The ADD_RULE function allows the user to add a new rule specifying existing facts and rule actions. The function checks that the rule does not already exist, and that all specified facts and rule actions are known. If any are not known, the user is informed.

The LINK_RULE function allows the user to link existing facts and rule actions to existing rules. In the present implementation no logic checking is done to the rule at this stage.

The RUN function with the present prototype carries out a forward chaining inferencing procedure within the dictionary environment to find what rules can fire, what actions will be obeyed and the changes in the fact profile for each inference step, for any given starting fact profile. Because of the dictionary formalism, the RUN function provides a basis for far more than a conventional rule trace.

The WHY_NOT function, for example, may be used to query why a rule did not fire in the inferencing process, in either all or a specified inference step. *Why_not* also provides for an examination of any rules not appearing in this set because they did not match the fact profile in any way. Figure 4.1 shows a sample output. As shown the user can be informed of exactly why a rule did not fire, either in a specified inference step or in all steps.

This more detailed display is possible by treating the rule as data and storing the

[1]Although not discussed here, all of the supplied functions can be represented in a query language formalism, for example SQL. See Jansen&Compton 88 for examples .

data in the relational model formalism. Once we can determine the fact profile current at an inference step, and are given the name of the rule to be explained, the step of determining which facts were missing involves simple table searching to extract the fact names and comparison with the fact profile applicable. A conventional rule trace is of course also available.

```
?-why_not '21500.39H'.

The rule 21500.39H was not able to fire in inference step 1
as none of the facts matched the fact profile current then.

The rule 21500.39H could not fire in inference step 2 because
it did not match the fact profile. The differences were :-

        vhthy was not asserted
        t3_not_missing was not asserted
        sick_euthy asserted but not used in rule
        comment_thyroid_surgery asserted but not used in rule
```

Figure 4.1 - WHY_NOT function

The DISPLAY_FACT_PROFILE function may be used to display the fact profile current at an inference step, as shown in figure 4.2.

```
?-display_fact_profile all.

At step 1 the fact profile was as
follows :-
        sick_euthy
        comment_thyroid_surgery

At step 2 the fact profile was as
follows :-
        sick_euthy
        comment_thyroid_surgery
        surgery
```

Figure 4.2 - DISPLAY_FACT_PROFILE Function

5. FUTURE ENHANCEMENTS

5.1 Context

The dictionary approach allows for full documentation of all the knowledge and provides a basis for setting up techniques for unrestricted browsing of the knowledge.

For example, the most difficult maintenance problem occurs when the expert system has failed to diagnose a specific case, although it knows a lot about similar cases. A new rule will be very likely to subsume old rules, and the best strategy is to modify an existing rule. The *Knowledge Dictionary* approach would be to do a USAGE enquiry for the rule action desired, and for key facts in the fact profile providing a *context* for the maintenance process. The subset of rules generated (or *context*) could then be examined with the WHY_NOT function after an inference run to find the best candidate for maintenance.

We hypothesize that the most useful feature will be flexible alteration of the *context* in which the knowledge is examined. The user may start with a broad fact profile, and subsequently narrow the profile by the addition of extra facts, thereby reducing the candidate rule they wish to inspect. If this addition were done in the context of the existing facts and associated rule premises, then an orderly narrowing of the fact profile is possible, without relying on the knowledge of the expert in the respective domain and the experience of the knowledge engineer in how the system's knowledge has been organized. This is in strong contrast to current tools. Context is similar to the concept of constraints or views in database theory.

Contextual availability of the knowledge can be of benefit in several areas. Firstly, the *Knowledge Dictionary* model allows the user to attach any number of classification terms to knowledge domain object occurrences, and these classifications, if applied to rules, facts, or rule actions may be used to further narrow the search area of candidate rules. This is analogous to the *grouped-by* operation implemented in relational theory.

Secondly, when adding rules, or adding new elements, either facts or rule actions, to existing rules, the user should be warned if the addition results in any conflict with existing contextual knowledge established by the currently known

relationships between facts, rule actions and rules[1]. This is not necessarily an error condition, as the new information may expand the currently known horizons.

We suggest that such a context facility will not only facilitate knowledge engineering and maintenance, but is a fundamental feature of human knowledge and therefore must be included in any artificially intelligent system. We have argued elsewhere (Compton and Jansen 88 & Compton *et al* 88) that experts never give the reason why they reach an interpretation, rather they *justify* why they are correct. Hence the rules that expert systems are composed of are based on these justifications. Such justifications are not absolute but depend on the context in which the knowledge is required. This is an example of Karl Popper's hypothesis (Popper 85) that human knowledge proceeds not by proving hypotheses are right, but disproving the alternatives, necessarily a small set dependant on the context.

5.2 Run Time System Generation

The *Knowledge Dictionary* provides an alternate representation for the knowledge within the domain of discourse. It is anticipated that for all but the most trivial systems, this representation, although inferencable in its own right, will result in unacceptable response times for a routine production system.

It will be required that the user be able to generate a run time version of the expert system from the *Knowledge Dictionary* representation[2]. The major advantages in having this function are that the user need not be capable of actually coding whatever the selected run time formalism required, and maintenance need only occur on the knowledge representation within the *Knowledge Dictionary* environment.

Currently, the *Knowledge Dictionary* can generate the required formalism to run the associated inference engine, but further work needs to be done to allow a general generate function, whereby the user indicates the formalism required, and the appropriate generation occurs.

6. CONCLUSIONS

The work to date suggests that the *Knowledge Dictionary* technology is suitable for use on expert systems, and that similar benefits are to be found as for conventional data processing systems. The decomposition of heuristics and their storage in the relational data model makes available the power of the relational data model for knowledge maintenance and browsing. If the system is small enough, then the dictionary environment may be suitable in its own right as an expert system shell, otherwise the facility of generating the run time formalism enables all maintenance to be carried out within the *Knowledge Dictionary*, and then reflected in the run time system when the knowledge engineer is certain the maintenance was successful.

We have shown that heuristics may be decomposed into constituent parts, facts and rule actions. In doing so, any heuristic is implemented in its most primitive form, and automatically documented and cross referenced. Using the dictionary environment, the documentation and cross referencing is automatically kept up to date, in a form understandable to both knowledge engineers and knowledge domain experts.

The decomposed form of the knowledge is inferencable using a simple inference engine obeying simple data manipulation rules.

We have highlighted a major feature of this approach. It provides a basis to flexibly alter the context enabling the knowledge base to be accessed with equivalent flexibility to human knowledge. This facility enables the expert system to emulate the domain expert's reasoning process more closely, and should aid in the knowledge acquisition

[1] This function is related to rule subsumption, conflict etc.

[2] This function should be similar to that found with fourth generation programming systems, where after describing the application in detail, the generation of the run-time code is automatic and generally algorithmic.

process by relaxing the Feigenbaum bottleneck.

7. ACKNOWLEDGEMENTS

This project would not have been possible without the contributions of the following, ranging from the initial development of GARVAN-ES1, to important advice for the current project. Bob Colomb and Neil Ashburner of the CSIRO Division of Information Technology. Kim Horn of Telectronics (formerly at the Garvan). Ross Quinlan of the University of Sydney. Leslie Lazarus, Ken Ho, Rick Symons, Ross Vining and other domain experts at the Garvan Institute.

8. REFERENCES

Al-Zobaidie A & Grimson J B, *Expert Systems and Database Systems: How Can They Serve Each Other?*, Expert Systems, February 1987, Vol. 4, No. 1.

Bachant Judith & McDermott John, *R1 Revisited: Four Years in the Trenches*, The AI Magazine Fall 1984.

Carlis J V, *HAS, A Relational Algebra Operator, or Divide is not Enough to Conquer*, IEEE International Conference on Data Engineering, Los Angeles, 1986.

Clancey William J, *Heuristic Classification*, Artificial Intelligence 27 (1985).

Codd E F, *A Relational Model for Large Shared Data Banks*, CACM, Vol. 13, No. 6, 1970.

Compton Paul& Jansen Bob, *Knowledge in Context: A Strategy for Expert System Maintenance*, CSIRO Division of Information Technology technical report, TR-FC-88-07, August 1988.

Compton Paul, *Expert systems for the clinical interpretation of laboratory reports.*, In: Van der Heiden O, den Boer NC, Souverijn JHM, eds. CLINICAL CHEMISTRY - Proceedings of the XIII International Congress of Clinical Chemistry: Plenum Press, 1987:in press

Compton Paul, Horn Kim, Quinlan Ross, Lazarus Leslie, Ho Ken, *Maintaining an Expert System*, Proceedings of the Fourth Australian Conference on the Applications of Expert Systems, Sydney 1988.

Debenham John, *Expert Systems: An Information Processing Perspective*, Proceedings of the Second Australian Conference on Applications of Expert Systems, May 1986.

Debenham John, *Knowledge Base Design*, The Australian Computer Journal, Vol 17, No. 1, February 1985.

Dolk Daniel R and Kirsch II Robert A, *A Relational Information Resource Dictionary System*, Communications of the ACM, January 1987, Vol. 30, Number 1.

Duda Richard O, Hart Peter E, Reboh Rene, Reiter John, Risch T, *SYNTEL : Using a Functional Language for Financial Risk Assessment*, IEEE Expert, Fall 1987.

Held James P & Carlis John V, *Conceptual Data Modelling of an Expert System*, Proceedings, the 4th International Conference on Entity-Relationship Approach, October 1985, IEEE

Horn K A, Compton P, Lazarus L & Quinlan R, *An Expert System for the Interpretation of Thyroid Assays in a Clinical Laboratory*, The Australian Computer Journal, Volume 17, No 1, February 1985.

Ishikawa H, Izumida Y, Yoshino T, Hoshiai T, Hoshiai A, *A Knowledge Based Approach to Design a Portable Natural Language Interface to Database Systems*, IEEE Proceeding, Conference on Data Engineering, Los Angeles 1986.

Jansen Bob & Compton Paul, *The Knowledge Dictionary: A Relational Tool for the Maintenance of Expert Systems*, CSIRO Division of Information Technology, technical report TR-FC-88-01, February 1988

Jansen Bob, *A Data Dictionary Approach to the Software Engineering of Rule Based Expert Systems*, AI'87 Australian Joint Artificial Intelligence Conference Proceedings, Sydney 1987.

Jansen Bob, *The Knowledge Dictionary: Integrating a Knowledge Base with a Data Store*, CSIRO Division of Information Technology technical report TR-FC-88-02, June 1988

Leung C M (Ricky) and Nijssen G M (Shir), *Database Oriented Expert Systems*, AI'87 Australian Joint Artificial Intelligence Conference Proceedings, Sydney 1987.

Popper Karl R, *Conjectures and Refutations The Growth of Scientific Knowledge*, Routledge and Kegan Paul 1985