Gaines, B. R. (1991). Induction and visualization of rules with exceptions. In Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop Boose, J. H. and Gaines, B. R., eds., pp. 7-1-7-17. University of Calgary, Calgary.

INDUCTION AND VISUALIZATION OF RULES WITH EXCEPTIONS

Brian R Gaines

Knowledge Science Institute
University of Calgary
Calgary, Alberta, Canada T2N 1N4
gaines@cpsc.ucalgary.ca

Abstract: The ripple down rule schema has been embedded within an integrated knowledge acquisition architecture such that: the induction module induces ripple down rules; the evaluation module checks them against large data bases, whether induced or manually generated/edited; the visualization module represents them graphically; and the knowledge representation server supports inferences based on them. In extending the system, ripple down rules have been fully integrated with the existing rule exception structure, and have themselves been extended to support multiple conclusions and multiple exceptions.

INTRODUCTION

There are now a number of significant examples of real-world applications of knowledge-based systems on a routine basis, and we are beginning to see the development of a second-generation literature that reports on longitudinal studies of practical experience, and on the rational reconstruction of working systems. Such reports are of great significance to the scientific advance of knowledge acquisition theories, methodologies and tools, since they provide a rich source of well-grounded experimental material. In particular, they present a challenge to ongoing research on knowledge acquisition to demonstrate that 'new' approaches are able to cope well with the data and applications involved, and, hopefully, provide insights and tools that can improve on what has already been achieved.

An important development in this area has been the dissemination of material by Paul Compton and various co-workers on the knowledge-based system for diagnosis of thyroid disorders developed at the Garvan Institute (Horn, Compton, Lazarus & Quinlan, 1985; Compton & Jansen, 1990a,b). The system is of particular interest because:

- It is in routine clinical use.
- Data on some 20,000 patients is available.
- The data is, in major part numeric, giving scope for inductive estimation of critical values.
- Conversion to symbolic form is available.
- The original rule base developed is available.
- An improved rule base using 'ripple-down' rules is available.
- Data from this system has already been used in a number of knowledge-based system studies.

This paper reports on a group of studies which took this dataset as a challenge to the integrated knowledge acquisition architecture described at last year's meeting (Gaines, 1990) and attempted to:

- Compare machine induction of ripple-down rules with human generation of such rules.
- Achieve adequate speed in both induction and evaluation on large datasets to support collaborative development of the knowledge base through machine support of human expertise.
- Provide visualization of ripple-down rules as a natural extension to a graphic knowledge editor.
- Integrate the ripple-down rule technique with 'rules with exceptions schema' and extend it to cope with multiple conclusions and missing data ('open' cases).
- Develop a logical analysis of ripple-down rules and rules with exceptions.

RIPPLE-DOWN RULES

Compton and Jansen (1990a,b) introduced the ripple-down rule technique as an elegant methodology for the acquisition and maintenance of large rule-based systems. The underlying concept is that people cope with the acquisition and maintenance of complex knowledge structures by making incremental changes to them within a well-defined *context* such that the effect of changes is locally contained in a very well-defined manner. Standard production rule systems do not have this property. The modularity of the rules themselves is not reflected in a modularity of the consequences of changes to those rules. Small changes can lead, through complex interactions, to major effects, making the development and maintenance of rule-based systems far more difficult than it appears at first sight (Li, 1991).

One merit of frame-based techniques is that they provide means for managing rules by structuring them as 'demons' concerned only with entities that are instances of specific frames. The taxonomy of the frames then provides a way to conceptualize the activities of particular clusters of rules, and the relations between entities and their slot fillers provides a way to conceptualize the propagation of rules. Hence, it is of interest to compare solutions to the problem that modify the rule structure with those that manage the problem through frames.

The ripple-down rule technique creates a two-way dependency relation between rules such that rule activation is investigated only in the context of other rule activation. If the premise of a parent rule is true for a particular individual then, if has no dependents, its conclusion will be asserted for that individual. If, however, it has an 'if-true' dependent then that rule, and its dependents, will be tested, and the original conclusion will only be asserted if the premises of none of them are true for the entity. Conversely, if the premise of a parent rule is false for a particular individual, then not only will its conclusion not be asserted but also, if it has an 'if-false' dependent then it, and its dependents, will be tested.

Thus, ripple down rules form a binary tree with two sorts of link between parent and sibling. Figure 1 shows six of the top level rules in Compton's reconstruction of the Garvan ES-1 diagnostic system as a set of ripple-down rules. There are 61 possible diagnostic combinations, labeled 00 to 60. The top level rule, 0.00, is an overall default with no conditions stating that is no other rule is activated the diagnosis is 00, no thyroid disorder. This rule is always true so it has only an 'if-true' link to rule 1.46 which gives four conditions leading to diagnosis 46. If these conditions are not met, the 'if-false' link to rule 2.32 is followed. If the conditions are met then diagnosis 46 is not immediately indicated. The 'if-true' link to rule 497.48 is first tested and if its conditions are satisfied its diagnosis of 48 takes precedence over 46.

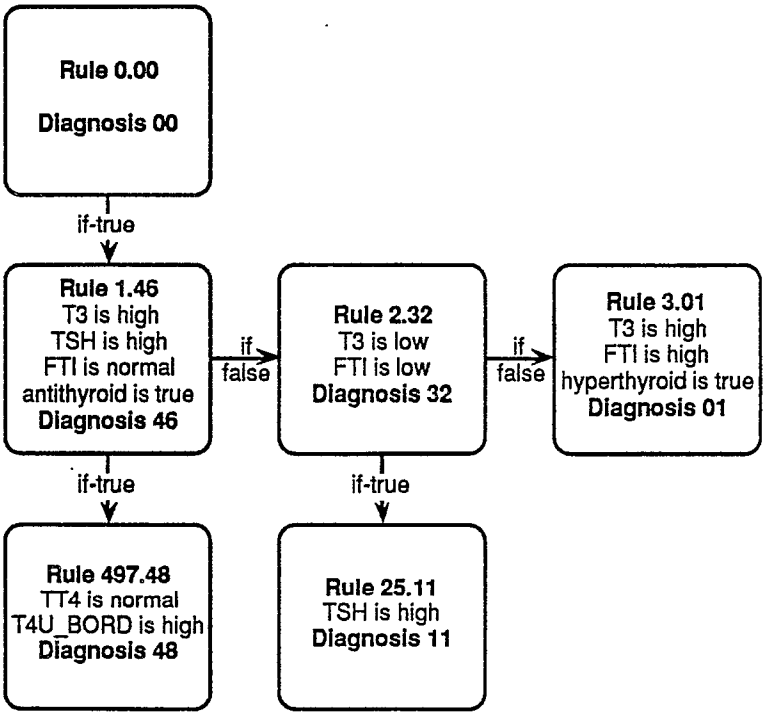

Figure 1 Some ripple-down rules

Logically, ripple down-down rules have exactly the same representational and deductive capabilities as standard production rules. Where they differ is in the strong context provided by the 'if-true' and 'if-false' links that guarantees that the effect of adding a rule is strictly contained in that it can affect the conclusions applied to cases previously falling under at most one other rule. A new sibling rule joined to a parent by an 'if-true' link can only change the conclusion applied to its entities falling under its parent, and no others. A new sibling rule joined to a parent by an 'if-false' link can only change the conclusion applied to its entities falling the rule with an 'if-true' link to the chain of which its parent is part, and no others. In general, a new rule can only affect cases falling under the rule responsible for the conclusion if the new rule were not present. This limits the comparisons and decisions required in knowledge acquisition and maintenance to the consideration of cases falling under only one rule.

For example, consider a situation in which a rule network has been built as shown in Figure 1, and a new case is under consideration for which rule 0.00 succeeds, 1.46 fails, 2.32 succeeds, and 25.11 succeeds. The conclusion is diagnosis 11, but suppose the correct diagnosis is 15. Since 25.11 succeeds but its conclusion is wrong, it is necessary to add a new rule with an 'if-true' link to 25.11. How should its premise be specified? Figure 2 gives the description of relevant cases available to the knowledge engineer. Case 25 is the one case that has so far been diagnosed using rule 25.11, and case 41 is the new case. In this table, the values of attributes satisfying the premises of the relevant rules are italicized, and the values in which two comparative cases differ are in bold. In considering the premises for a new rule, one is aiming to create one such that case 41 is included so that it can be diagnosed as 11, but case 25 is excluded so that it remains diagnosed as 11. Hence, the relevant differences are that *sick* is true for 41 and false for 25, *ovulatory* is false for 41 and true for 25, *TSH_BORD* is high for 41 and missing for 25. One could generate a very specify rule requiring all three differences to be present, or a more general rule requiring just one or two to be present. In this case the *sick* attribute was chosen as the differentiation resulting in the 'if-true' addition of rule 41.15 shown in Figure 3.

Now consider case 79 for which rule 0.00 succeeds, 1.46 fails, 2.32 succeeds, and 25.11 fails. The conclusion is diagnosis 32 from the last successful rule, but suppose the correct diagnosis is 01. Since 2.32 succeeds but its conclusion is wrong, it is necessary to add a new rule with an 'if-false' link to 25.11. How should its premise be specified? In figure 2, Case 2 is the one case that has so far been diagnosed using rule 2.32, and case 79 is the new case. One is aiming to create one such that case 79 is included so that it can be diagnosed as 01, but case 2 is excluded so that it remains diagnosed as 32. Hence, the relevant differences are that *hypothyroid* is true for 79 and false for 2, *ovulatory* is true for 79 and false for 2, *TSH* is missing for 79 and normal for 2, *T3_BORD* is missing for 79 and low for 2, *FTI_BORD* is low for 79 and missing for 2. In this case the *TSH, hypothyroid* attributes were chosen as the differentiation, and the *sick* attribute was also included resulting in the 'if-false' addition of rule 79.14 shown in Figure 3.

Note that the addition of a non-differentiating attribute causes no problems, providing at least one differentiating one is used. It is meaningful in that it will affect whether future cases fall under rule 79.14 or not.

This constrained but indeterminate choice of conditions for the premise of a new rule is the main feature of ripple down rule acquisition and maintenance. The expert, or knowledge engineer, has only to consider a limited and well-defined set of cases for which conclusions have been drawn. The possible clauses for the premise of a new rule are constrained to cover the new case but not the previous cases. Within these broad constraints, the expert or knowledge engineer has the discretion to use their knowledge to design as specific or as general a rule as they deem appropriate. Thus, the ripple down rule structure allows the computer to support the management of large rule bases while leaving the human designer the capability to use his or knowledge effectively in a well-defined decision framework.

	2	25	41	79
sex	female	female	female	female
on_t4	false	false	false	false
query_t4	false	false	false	false
antithyroid	false	false	false	false
sick	true	false	***true***	***true***
pregnant	false	false	false	false
i131	false	false	false	false
hypothyroid	false	false	false	**true**
hyperthyroid	false	false	false	false
lithium	false	false	false	false
tumour	false	false	false	false
goitre	false	false	false	false
hypopit	false	false	false	false
psych	false	false	false	false
antibs	false	false	false	false
surgery	false	false	false	false
treated	false	false	false	false
no_comment	true	true	false	false
screening	false	false	false	false
ovulatory	false	true	**false**	**true**
TSH	normal	*high*	*high*	***missing***
TSH_BORD	missing	missing	**high**	missing
T3	*low*	*low*	*low*	*low*
T3_BORD	low	missing	missing	**missing**
TT4	missing	missing	missing	missing
TT4_BORD	missing	missing	missing	missing
T4U	missing	missing	missing	missing
T4U_BORD	missing	missing	missing	missing
FTI	*low*	*low*	*low*	*low*
FTI_BORD	missing	missing	missing	**low**
TBG	missing	missing	missing	missing
TBG_BORD	missing	missing	missing	missing
diagnosis	32	11	15	14
RULE	2.32	25.11	41.15	79.14

Figure 2 Some cases relevant to these rules

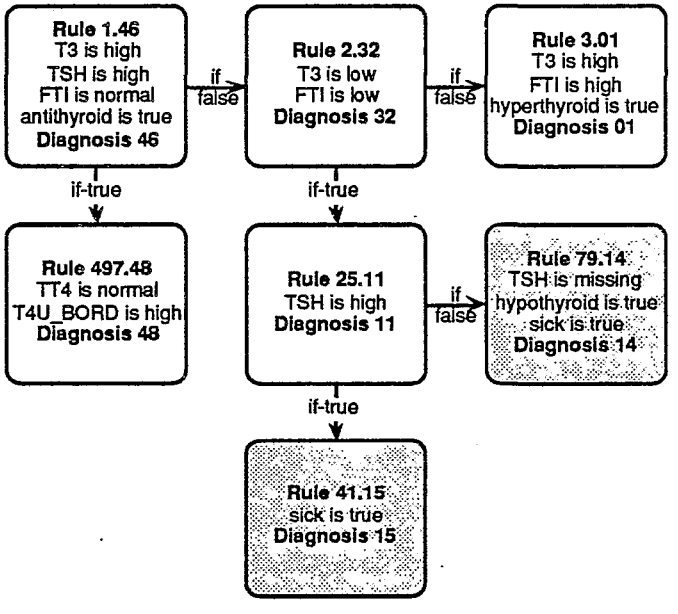

Figure 3 Ripple-down rules after incremental acquisition

There are three extensions to the situation illustrated that require further discussion. First, the new cases may be identical in features to an old case but have a different diagnosis. The cases cannot be differentiated and the expert or knowledge engineer has to cope with the indeterminacy, perhaps by gathering more data if there are missing values in the new case, perhaps by accepting the indeterminacy and giving alternative diagnoses, possibly with probability estimates. Clearly, this situation is outside the scope of a deductive process and requires a different solution to those under consideration.

Second, there are multiple cases falling under the existing rule, not just the single case so far exemplified. This causes no fundamental problems since, if the first situation does not apply, there will be at least one attribute differentiating the new case from any given previous case, and a combination of such differentiating attributes provides a rule differentiating it from all the previous cases. Hopefully, the situation simplifies, in that one attribute is sufficient to differentiate several previous cases, but even if it does not there is always a solution available.

Third, the expert or knowledge engineer may chose to not differentiate all previous cases in the multiple case situation. In this event an additional 'if-true' rule is required as a sibling of the new rule to cover the old cases that have been allowed to fall under the new rule. This would be done because the pair of rules generated in this way were perceived to be simpler than a single differentiating rule.

EXPERIENCE WITH RDRS AND DIRECTIONS FOR FURTHER RESEARCH

The utility of the ripple down rule approach has been demonstrated in large scale applications. The 650 complex rules of the Garvan ES-1 system have been reduced to some 550 simpler ripple down rules, and the rate of developing the rule base has been increased from about 2 rules a day to 10 rules an hour (Compton and Jansen, 1990b). Compton, Edwards, Lazarus, Malor, Menzies, Preston, Srinivasan & Sammut (1991) report a new successful application to chemical pathology at St Vincent's hospital.

The Garvan data were used some time ago for experiments in inducing rules using the system that was eventually issued as C4.5 (Quinlan, Compton, Horn & Lazarus, 1987), and Mansuri, Kim, Compton and Sammut (1991) have reported comparisons between ripple-down rules, ID3 and C4.5 on a study based on some 9,500 Garvan cases. In general, the number of rules generated by induction is similar to that generated manually through ripple down rules, as are the error rates. However, the induction tools used in these studies do not themselves generate the exception structure of ripple-down rules, and it would be useful to know how an induced set of ripple down rules compares with manually generated rules, in numbers, errors rates, and comprehensibility as knowledge structures.

The ripple down rule technique not only supports the incremental modification of the rule structure but also has a natural graphic knowledge representation as is apparent in Figures 1 and 3, and it would be useful to portray this in a knowledge visualization tool. This might aid understanding of the, still significantly large, knowledge structures generated, and also might support the global reconstruction of the ripple down rule tree, for example when over-specialization in earlier rules has caused a greater proliferation of later rules than is necessary.

As already noted, ripple down rules do not change the representational or deductive capabilities of the inference system. Hence, one now has three representational schema for rule-based systems: standard production rules without exceptions; rules with exceptions such that if the premise of one is satisfied it inhibits others (Gaines, 1991b); and ripple down rules with two forms of exception as described above. A multiplicity of representations may be valuable in knowledge representation in allowing a knowledge structure to be described in a way that is natural to human experts, or it may represent 'over-engineering.' Hence, it is interesting to investigate the relation between these three rule schema, particularly their inter-translation, and also to see whether they can be integrated into a single schema.

The ripple down rule schema as described depends on there being only one conclusion applicable to a particular case. It is often possible to restructure diagnostic situations such that this is so (about half the 61 Garvan diagnoses are compounds of multiple basic diagnoses), but this may be detrimental to natural representation and induction, and it is not always possible in more general systems. Hence it is interesting to investigate the extension of the ripple down rule schema to allow for multiple conclusions, and the possibility of embedding several such schema in a single knowledge structure concerned with multiple classifications.

These topics are the concern of the remainder of this paper. The ripple down rule schema has been embedded within the KSSn architecture (Gaines, 1991a) such that: the Induct module induces ripple down rules; the evaluation module rapidly checks them against large data bases, whether they are induced or manually generated/edited; the KDraw visualization tool represents them graphically; and the KRS knowledge representation server supports inferences based on them. In extending the system, the ripple down rules have been fully integrated with the existing rule exception structure, and have themselves been extended to support multiple conclusions and multiple exceptions. Since previous papers on KSSn have not focused on its inductive component, the next section will describe this independently of ripple-down rules.

KSSn INDUCTIVE MODELING SUB-SYSTEM

Inductive modeling of large databases is now an attractive proposition given the availability of fast empirical induction tools that can cope with the processing of large databases with many fields and data types (Piatetsky-Shapiro & Frawley, 1991). However, while the rules output from an induction tool may be capable of providing decision support when loaded into an expert system shell they have major weaknesses as 'knowledge', notably that they are difficult for people to understand, to some extent individually, and certainly in terms of their interactions.

An induction tool used on a database of a few thousand cases may sensibly generate several hundred rules in a few minutes. These rules may perform well in predicting one attribute from others but they will usually be almost impossible for someone to understand without extensive analysis. Full understanding probably only comes with editing and re-evaluation, "This rule makes more sense if I change it in this way. This block of rules is probably better thought of in this way. How am I doing with my 'improvements'?" There is a need for output modules that help people navigate through the results of empirical induction, seeing them in the most understandable form, and being able to edit and refine them to become humanly meaningful knowledge while continually checking that performance based on the revised rules is not deteriorating.

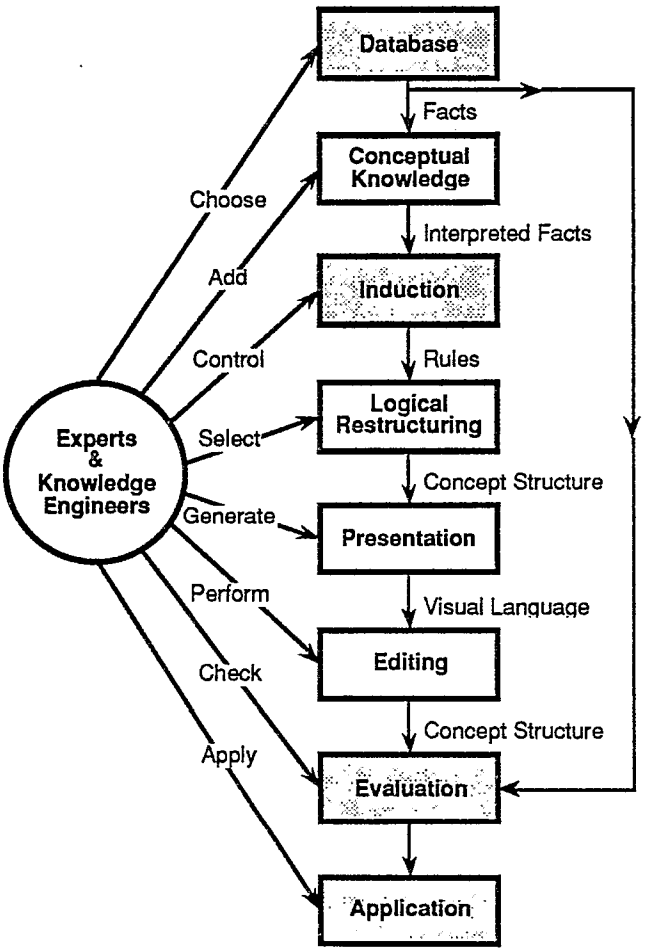

Figure 4 Induction process in KSSn

The design objectives for the induction functionality of KSSn are illustrated in Figure 4. A community of domain experts and knowledge engineers is attempting to extract knowledge from a large database. The 'community' may range from one individual to a large team, and its expertise may range from very little in an exploratory study to a great deal in a highly targeted analysis. The expertise may be assumed to change with the acquisition process so that, for example, highly automatic use of tools may be important in the early phases but detailed human control may become significant in later phases.

A stereotypical induction tool consists of the four shaded boxes in Figure 4:

Database→Induction→ Evaluation→Application

The additional interpolated boxes have been added to provide additional functionality to cope with some of the requirements noted above.

There is a need to supplement most databases with *conceptual knowledge* prior to induction because this is often implicit in the use of the database or just totally missing. However, its availability is important both to the effective use of the data available since the re-induction of already known concepts is wasteful of data, and detrimental to the quality of the results since the addition of a conceptual framework usually leads to increased generality and more meaningful interpretation. Often, the additional concepts involve ranges of numbers or grouping of categories. For example, in a medical database one may know that children in the age range 0-6 form a particular group at risk with respect to some diagnostic category and enter the conceptual classification that $0 \leq age \leq 6$ corresponds to "very young." This pre-classification will greatly reduce data requirements if it is appropriate (Quinlan, Compton, Horn & Lazarus, 1987). If it is not appropriate, given enough data, the induction algorithm will generate a more effective range. Núñez (1990) has shown that adding conceptual knowledge can be very effective in decision tree induction in producing more effective and smaller trees with less data. In general, introducing conceptual knowledge prior to induction is important in allowing the *data–knowledge* trade-off relation (Gaines, 1991d) to be manipulated effectively. It integrates the knowledge acquisition techniques of 'expertise transfer' and 'empirical induction'.

The *logical restructuring* of rules produced by an induction tool is also very important. The same logical structure can be manipulated to many different forms, some of which are very much more understandable than others. One example is the human tendency to conceptualize in terms of stratified systems of rules with multi-level exceptions (Compton & Jansen, 1990). For example, "a purchase order from a subsidiary to the main company should have order codes 08 or 09 except if that subsidiary is a manufacturing or packaging plant in which case the order code should be X5K or 113 unless the plant is overseas in which case the order code should be X5J." Such ramified structures of rules and exceptions are common in the constraints applied to large corporate database use, and their expression as an equivalent set of flat rules would be scarcely recognizable to users. Another example is the factoring out of common clauses in complex rules to form named sub-concepts, for example, "low quality supplier" for a supplier who is financially weak and has erratic delivery. Some degree of automatic restructuring is possible to minimize the expression of a rule set by using exceptions and factoring. However, minima are usually neither unique nor necessarily the most understandable expressions.

The *presentation* of the resultant concept structures is important to human understanding and evaluation as knowledge. Nosek and Roth (1990) have demonstrated empirically that the visual presentation of knowledge structures as semantic nets leads to more effective human understanding than does textual presentation of the same structures. Visual languages for knowledge representation can have well-defined formal semantics that are inter-translatable with standard textual representations (Gaines, 1991c). Hence, they may be treated as alternative ways of expressing a knowledge structure with, perhaps, the experts using one format and the knowledge engineers another.

Whatever form of presentation is adopted it is important that the users be able to *edit* in that format also. As already noted, some editing will inevitably be required and it is extremely frustrating to have to switch representations to make changes in the knowledge structures. Thus, a visual language should be implemented through a structure editor providing simple and natural interaction, not as a presentation mechanism only.

The results of editing should then be subject to *evaluation* as simply and directly as possible. For large datasets it may not be feasible to provide interactive evaluation in which the performance changes resulting from editing the knowledge structures are immediately displayed, but this is an important long-term objective as technology improves. Minimally, a fast evaluation against specified data with a display of the results in terms of the knowledge structure being edited is required. That is, the user should be able to see clearly and explore easily the sources of errors in terms of concepts, rules and cases, and their effects on overall performance parameters.

Figure 5 shows the architecture of an induction system developed to support the processes of Figure 4. The user interface for both the prior introduction of conceptual knowledge and the presentation and editing of the results of induction is a graphic knowledge editor for a term subsumption knowledge representation language (Gaines, 1991c). This enables the original database description to be expressed visually and then enhanced through the development of additional concepts and a concept hierarchy. The classification capabilities of the associated knowledge representation server are then used to extend the dataset with a set of binary attributes corresponding to a case falling under a concept or not.

The extended dataset is used as input to an induction engine that produces a minimal set of rules predicting specified attributes (which may include defined concepts). One option of the induction process is to produce stratified rules with multi-level exceptions (Gaines, 1991b). Biases may be introduced in the process through preferences for what values of attributes are taken to be defaults and what to be exceptions. The concepts forming the premises of rules can be factored to minimize the knowledge structure.

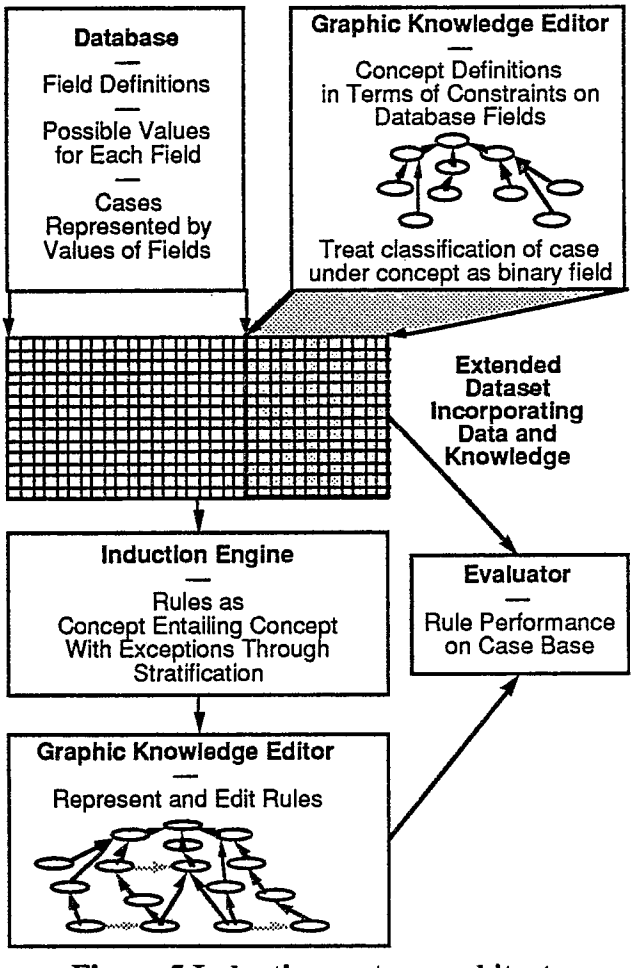

Figure 5 Induction system architecture

The resultant knowledge structure is presented visually in the graphic knowledge editor as an extension to the original structure entered. It may be freely edited and re-evaluated in terms of overall performance, and that of individual rules, against the original dataset and additional test data. Thus, from the perspective of the users, the flow of operations shown in Figure 4 can be carried out within the architecture of Figure 5. The components of Figure 5 are a set of modules around a knowledge representation server. This makes it feasible to integrate knowledge acquisition from databases with that from other sources such as text and human experts.

The system is best illustrated in operation through an example. Figure 6 is a dataset presented by Michie (1989) as a significant problem for NASA that was solved through automatic rule induction. The problem is one of deciding whether to use the shuttle autolander given various environmental conditions. The dashes in the dataset are *don't care* values. Their prevalence shows that it is a set of stereotypes rather than factual cases, and hence already contains additional knowledge. There would be 256 cases in the table if all the *don't care* values were instantiated. The elision of attributes in this way is a significant mechanism for the expression of expertise, as Bareiss (1989) showed in the PROTOS audiology study, and it important that an induction tool be capable of treating it correctly.

Case	stab	errors	sign	wind	mag	vis	class
1	-	-	-	-	-	no	use auto
2	no	-	-	-	-	yes	not auto
3	yes	lx	-	-	-	yes	not auto
4	yes	xl	-	-	-	yes	not auto
5	yes	mm	negative	tail	-	yes	not auto
6	-	-	-	-	out	yes	not auto
7	yes	ss	-	-	light	yes	use auto
8	yes	ss	-	-	med	yes	use auto
9	yes	ss	-	-	strong	yes	use auto
10	yes	mm	positive	head	light	yes	use auto
11	yes	mm	positive	head	med	yes	use auto
12	yes	mm	positive	tail	light	yes	use auto
13	yes	mm	positive	tail	med	yes	use auto
14	yes	mm	positive	head	strong	yes	not auto
15	yes	mm	positive	tail	strong	yes	use auto
16	yes	mm	negative	head	-	yes	not auto

Figure 6 Exemplary dataset for shuttle autolander decision (from Michie, 1989)

Figure 7 shows a set of rules derived by expressing as rules the decision tree given by Michie as a solution to the problem of predicting the final attribute "class" from the others. As Cendrowska (1987) has noted, the direct conversion of a decision tree to rules leads to some redundancy. However, those shown serve to exemplify the problem of comprehension when raw rule sets are presented to experts.

vis = no : *use auto*
vis = yes & errors = ss & stab = yes & mag = light: *use auto*
vis = yes & errors = ss & stab = yes & mag = med: *use auto*
vis = yes & errors = ss & stab = yes & mag = strong: *use auto*
vis = yes & errors = ss & stab = yes & mag = out: *not auto*
vis = yes & errors = ss & stab = no: *not auto*
vis = yes & errors = mm & stab = yes & sign = negative: *not auto*
vis = yes & errors = mm & stab = yes & sign = positive & mag = light: *use auto*
vis = yes & errors = mm & stab = yes & sign = positive & mag = light: *use auto*
vis = yes & errors = mm & stab = yes & sign = positive & mag = strong & wind = head: *not auto*
vis = yes & errors = mm & stab = yes & sign = positive & mag = strong & wind = tail: *use auto*
vis = yes & errors = mm & stab = yes & sign = positive & mag = out: *not auto*
vis = yes & errors = mm & stab = no: *not auto*
vis = yes & errors = lx: *not auto*
vis = yes & errors = xl: *not auto*

Figure 7 Decision tree for dataset converted to rules without pruning

Figure 8 shows a minimal rule set, equivalence in performance to that of Figure 7 generated by Induct from the data of Figure 6. One would expect the smaller rule set in itself to be easier to understand.

Representation of the subsumption involved in the exceptions is natural in graphical form. The lower part of Figure 9 shows the rules of Figure 8 translated directly to the visual language of the graphic knowledge editor in KSSn. In the upper part of the figure the database fields and possible values are shown as a domain description. Note that duplicate graphic items are treated as semantically identical so that the exception arrows between rules are shown only once.

```
: not auto
    Except:—
    vis = no: use auto
    stab = yes & errors = ss: use auto
        Except:—
        vis = yes & mag = out: not auto
    stab = yes & errors = mm & sign = positive: use auto
        Except:—
        vis = yes & mag = out: not auto
        vis = yes & wind = head & mag = strong: not auto
```

Figure 8 Induct rule set with exceptions

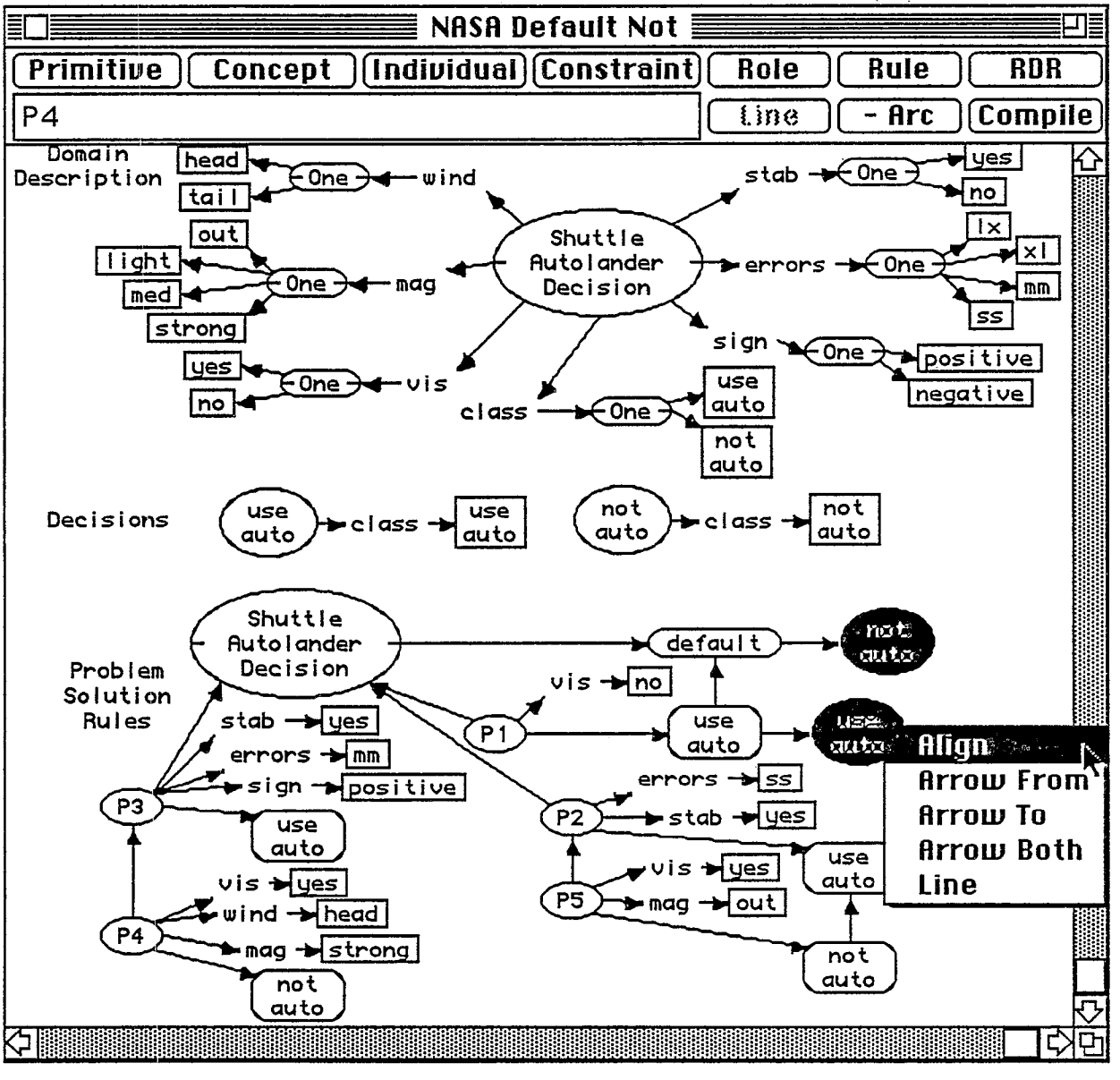

Figure 9 Database domain structure and rules in graphic knowledge editor

The visual language used in Figure 9 is precisely defined (Gaines & Shaw, 1990). Concepts are ovals, primitive concepts are ovals with small horizontal lines inside each side, individuals rectangles, roles unboxed text, rules rounded-corner boxes, constraint expressions rounded-corner boxes with small horizontal lines. Lines without arrows connecting primitive concepts denote that the concepts are disjoint, and connecting roles that they are inverse. The interpretation of the arrows in the editor is overloaded but well-defined by the types of the objects at their head and tail, e.g.:

concept → concept	definitional subsumption
concept → role → concept	definitional role with conceptual constraint
concept → role → constraint	definitional role with extension, cardinality or numeric constraint
constraint → individual	extensional constraint
individual → concept	asserted constraint on individual
individual → role → individual	asserted value of role for individual
concept → rule → concept	production rule
rule → rule	first rule exception to second

Human-computer interaction in the editor is modeled on Apple's *MacDraw* with additional features appropriate to the language such as arcs remaining attached to nodes when they are dragged. A popup menu as shown at the bottom right of Figure 9 allows connecting lines to be entered easily. The syntax of possible node interconnections and constraint expressions is enforced—it is not possible to enter a graph that is syntactically incorrect. Cut-and-paste of graphs and subgraphs is supported, and popup menus allow nodes to be connected with the minimum of effort. Updates are efficient and graphs with over a thousand nodes can be manipulated interactively. Scroll, zoom and fit-to-size facilities allow large data structures to be navigated easily. However, partitioning data structures over several screens is encouraged and has proved practical in managing large knowledge structures.

The user can evaluate the knowledge structure against specific cases visually or through automatic evaluation against cases stored in the knowledge representation server. He or she can compile the visual knowledge structures to the textual language of the server, or to a range of knowledge-based system shell languages, for evaluation on the original, or other, databases, and for application in a number of performance tools.

EXTENSION OF THE VISUAL LANGUAGE TO RIPPLE DOWN RULES

At first sight extending the visual language to encompass ripple down rules might seem obvious—add two arrows, one for 'if-true' and one for 'if-false', and then diagrams such as those of Figures 1 and 3 have a natural translation. The syntactic constraints are that: these arrows go only between rules; that each rule has at most one incoming arrow, and at most two outgoing arrows (one 'if-true' and the other 'if-false'); and that there are no loops. This approach leads to two questions: do either of the arrows correspond to the existing 'exception' arrow between rules; and does the syntax of multiple 'if-true' and 'if-false' arrows, into and out of rules have an unambiguous semantics capturing that of multiple conclusions and exceptions?

The semantics of the existing exception arrow between rules and that of the 'if-false' arrow in ripple down rules are the same. Their conceptualizations are different because the existing arrows are seen as inhibitors of the rules to which they connect, whereas the 'if-false' arrows are seen as passing on cases that do not fall under a rule. These two viewpoints are however complementary, and mutually consistent in the case of standard ripple down rules.

However, the extension of semantics based on two types of arrows to multiple 'if-true' and 'if-false' arrows entering and exiting rules breaks down. There is a natural disjunction for multiple 'if-true' arrows entering a rule—a case satisfying any of the rules initiating the arrows should be further tested. There is also a natural disjunction for multiple 'if-false' arrows entering a rule—a case satisfying any of the rules initiating the arrows should bot be further tested. However, if a mixture of arrows entered there is potential conflict and no natural resolution.

It turns out that the natural generalization is not to have two types of arrow, but rather two types of rule. One generalizes Compton's notion of *context* rather than that of 'if-true' and 'if-false' rules. The rule that an 'if-true' arrow enters marks the beginning of a new context established by the link from the rule that was true. All arrows from this rule propagate this context, and hence it is appropriate to mark the rule itself as being of a different type from the others. Figure 10 shows the rules of Figure entered in the grapher using only one arrow between rules but two different types of rule. An 'if-true' arrow is now one that enters a *context-marking* rule (with internal vertical bars at each end), hereafter termed an RDR. An arrow entering an ordinary rule is an 'if-false', or inhibitory connection. The semantics of multiple arrows in or out of rules is now unambiguously defined, and the only residual constraint is that there must be no loops.

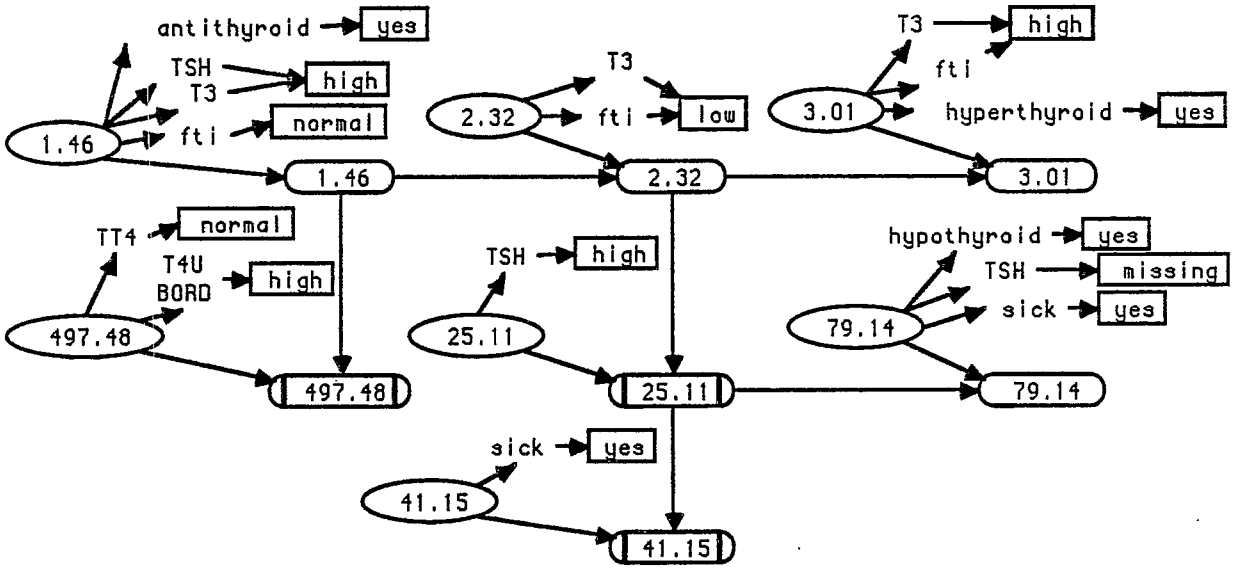

Figure 10 Ripple down rules of Figure 3 in KDraw

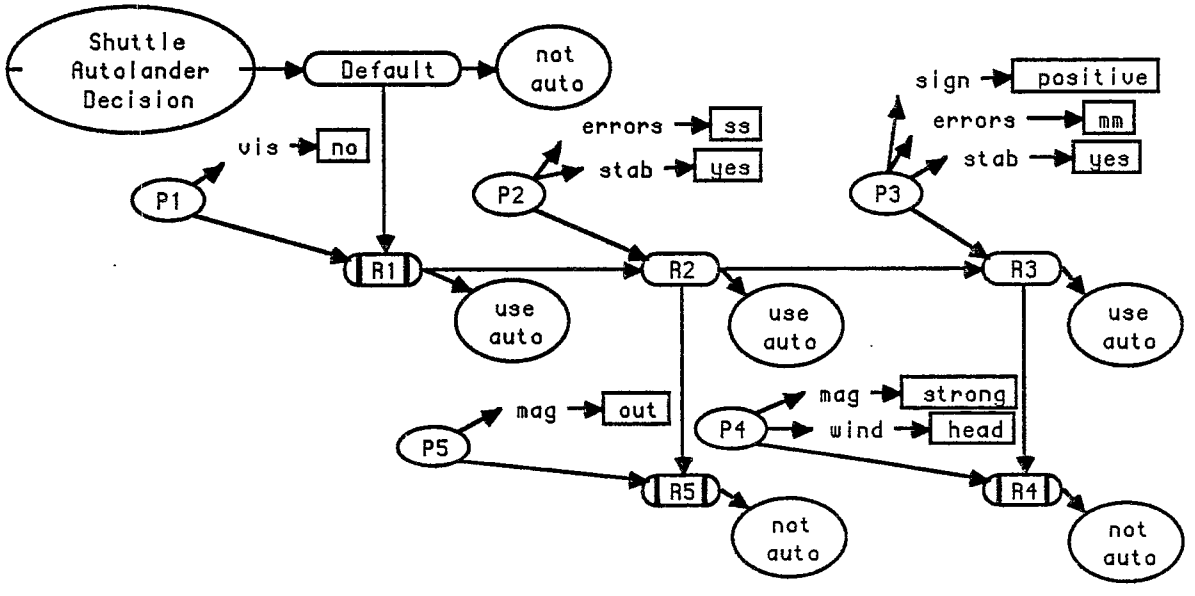

Figure 11 Ripple down rules for problem of Figure 6

Figure 11 shows ripple down rules induced by Induct for the shuttle autolander problem of Figure 6 with "not auto" as the default. In comparison with Figure 9, the subsumption structure of the premises is no longer overtly in the relations between the premise concepts, but is now in the relations between the rules. The structures are logically identical but their representational form is different. Figure 12 shows how the extended interpretation of the 'ripple down' arrows allows the solution to be formulated through 'if-true' arrows only, making the relation to Figures 9 and 11 apparent. Note the extra "vis = yes" clauses necessary in P4 and P5 that were implicit in the 'if-false' link from R1 in Figure 11. Note the additional 'if-true' link from R3 to the "mag = out" exception which was implicit in the 'if-false' link from R2 in Figure 11. The contexts are weaker in Figure 12 making the knowledge more explicit and the structure less compact.

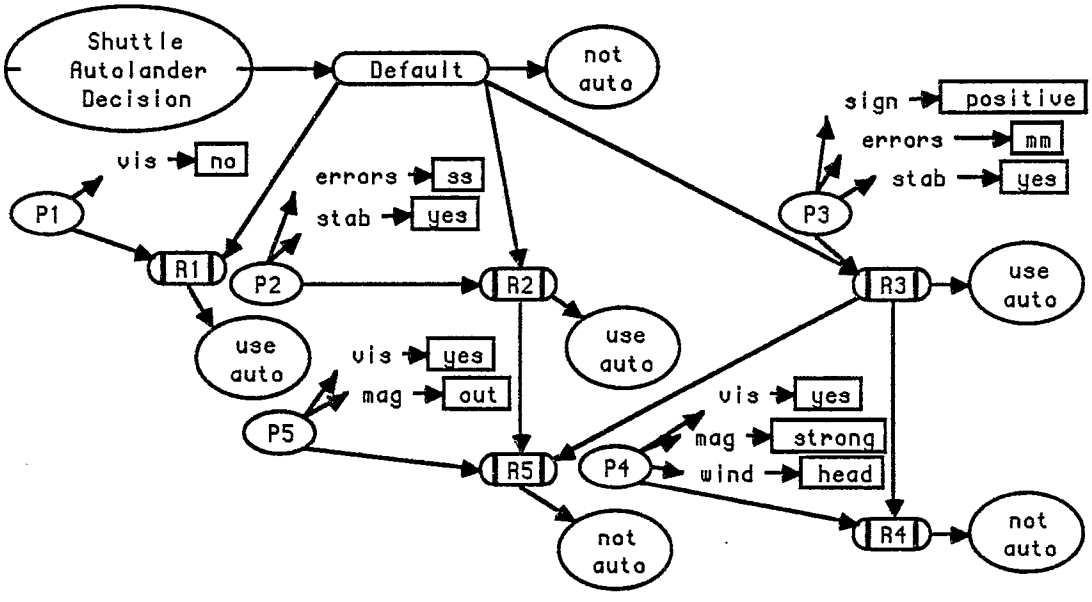

Figure 12 Variant rules for problem of Figure 6

Thus, as already noted, the two types of exception now allow a multiplicity of representations of the same knowledge structure. In Figure 9 there are only 'if-false', inhibitory links. In Figure 12 there are only 'if-true', excitatory links. In Figure 11 the two combine in a strict ripple down rule structure. Other combination are possible. Until there is further experience, probably of more complex situations with multiple conclusions possible for a single case, it is difficult to determine whether one has achieved 'representational richness' or degenerated into 'over engineering.' Currently, the naturality of KRS's original exception relation between rules, and the neat representation possible with ripple down rule structures as illustrated in Figures 10 and 11, suggests that the two forms of rule/link are useful knowledge constructs.

INDUCTION OF RIPPLE DOWN RULES

Inducing ripple down rules is very straightforward. The basic algorithm in Induct (Gaines, 1989) is to search for the premise for a given conclusion that is least likely to predict that conclusion by chance. Exception generation is automatic in that extending a premise with additional clauses to give fewer errors may also increase the probability that the improved result is due to chance—fewer cases predicted more accurately is not necessarily better on this measure than more cases predicted less accurately. When errors remain after the generation of a rule, Induct calls its rule generation algorithm recursively on the cases falling under the rule to generate exception rules for the error cases. This corresponds to the generation of the 'if-true' links in ripple down rules. To generate 'if-false' links, Induct calls the algorithm recursively for the residual cases that do not fall under the rule previously generated.

When Induct is run on the 9445 Garvan cases used in the UNSW studies of induction and ripple down rules, it generates 170 rules accounting perfectly for the dataset. This compares favorably with the some 550 ripple down rules generated in the reconstruction of the Garvan ES-1 using the same data. When Induct is run on the first 8000 cases and tested on the complete dataset it generates 172 rules and has an error rate of 0.4%. This compares favorably with the results reported by Mansuri, Kim, Compton and Sammut (1991) of: 550 rules, 0.5% errors for hand generated ripple down rules; 719 rules, 0.4% errors for ID3; and 515 rules, 2.1% errors for C4.5 with rule pruning.

Figure 13 shows the complete set of induced rules, diagnoses and number of cases falling under each rule. Apart from the difference in number, there is also a difference in structure from the manually produced set. The manual set has 41% 'if-true' links, and the mean number of clauses in a premise is 1.3. The induced set has 16% 'if-true' links, and the mean clauses is 3.8. It appears that rules in the manual set are over-general compared with those in the induced set and hence involve more correction rules.

It is reasonable to expect induction of ripple down rules to be substantially better than hand generation if we compare the circumstances of the human expert and the Induct algorithm. The expert and the algorithm are doing exactly the same thing at each node of the ripple down rule tree—that is adding additional rules to improve the performance. However, the expert is doing this in response to one erroneous case whereas the algorithm has available to it a complete analysis of the statistics of all relevant cases. Clearly, it is an empirical result as to how much better the algorithm will be, and the 3:1 improvement obtained for this dataset is only indicative of what might occur with other datasets.

There are a number of comments to be made on these results. First, that the decrease in the size of the rule set by no means detracts from the value of the manual acquisition of ripple down rules. At the start of a knowledge-based system development there may not be enough data available for induction, but there may be adequate human knowledge to build a working system. Later, when that system has been in use for some time, ripple down induction may be used to reduce the rule base without undermining the basic conceptual structures for use and maintenance to which those responsible for the system have become accustomed.

Second, there are more subtle reasons also why manual acquisition remains very important. These show up well in the recent St Vincent's studies reported by Compton, Edwards, Lazarus, Malor, Menzies, Preston, Srinivasan & Sammut (1991). The expert in entering new ripple down rules also enters new concepts. That is, the conceptual framework being applied to the cases is elaborated as part of the ripple down rule knowledge acquisition process. The contextual constraints already described continue to apply so that the use of new concepts can be managed locally and does not involve the expert in reanalysis of other parts of the rule tree. Some of the new concepts entered fall completely outside those already entered. Others are intervals on existing numeric data. The latter could be induced with sufficient data but the former could not. Other knowledge acquisition techniques, such as the repertory grid, might be used to elicit some of the additional concepts in advance, but there is already a strong case for manual ripple down rule elicitation as an effective acquisition technique for conceptual schema.

Third, the best features of the manual and inductive approaches can be combined. The evaluation tool in KSSn is fast enough for interactive use. It operates at some 200,000 rules a second on a Mac2ci and hence evaluates a rule set of some 170 rules on 9,500 cases in 8 seconds. An expert or knowledge engineer can make arbitrary changes to the ripple down rule structure and have a complete account of the effects rapidly enough to maintain reasonably interactive development. The Induct algorithm under the same circumstances takes 8 seconds on average to generate a new rule so that it can be used to advise the expert or knowledge engineer of an appropriate rule, which may then be manually amended, rather than generate all rules completely automatically.

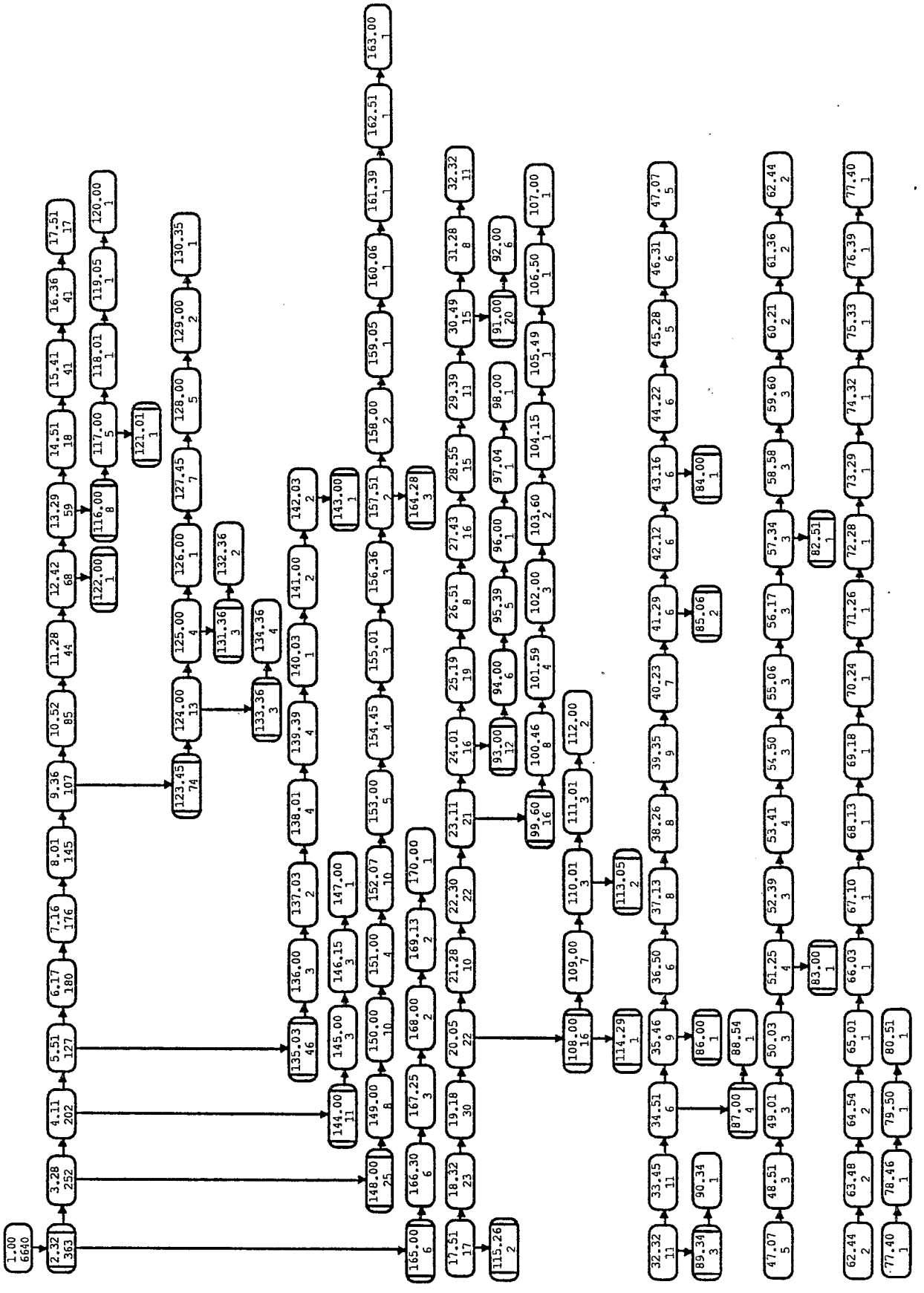

Figure 13 Ripple down rule structure induced for the Garvan data

CONCLUSIONS

This is a preliminary report on ongoing research. One can conclude so far that ripple down rules may be induced effectively and that they also integrate well with the visual representation of term subsumption knowledge representation schema. Extending and completing the system described to fully support mixed initiatives between manual and automatic knowledge modeling appears to offer significant advantages.

It will be interesting to determine whether the rule structure of Figure 13 is seen as a meaningful knowledge structure by those with relevant clinical experience. The top level structure in particular has large numbers of cases under rules with comparatively few exceptions and the rationale for each clause in the premise should be evident. Some of the exceptions correspond to the addition of diagnoses in a compound diagnosis which is clearly a valid reason for an exception. Others correspond to what may be variant diagnoses. The structure seems compact enough for a clinician to work through it methodically determining whether it is meaningful.

There is much that remains to be done with the Garvan dataset. Another 10,000 cases are available which will give some independent validation of the induced structures. There is adequate data to induce the numeric thresholds used in the conversion to symbolic form. It will be interesting to compare the detailed knowledge structures of the original rules, the manual ripple down rules and the induced rules. Paul Compton will be making the datasets and rules available as a Sisyphus project and it will be interesting to bring an increasing number of methodologies and tools within the same experimental framework.

It would also be very insightful to apply the same techniques to other datasets, notably the current St Vincent's study. It is not easy to find large-scale problems where both the data and expert knowledge are available to allow the types of methodologies and tools outlined in this paper to be properly evaluated. We need at least as much effort to be put into ongoing Sisyphus projects as is put into methodology and tool development. Knowledge acquisition needs to look to its empirical as well as its theoretical foundations.

ACKNOWLEDGEMENTS

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada. I am very grateful to Paul Compton who provided the inspiration for much of this work. In particular, access to his own research and the datasets on which it has been tested have been invaluable in the work reported here.

REFERENCES

Bareiss, R. (1989). **Exemplar-Based Knowledge Acquisition.** New York: Academic Press.

Cendrowska, J. (1987) An algorithm for inducing modular rules. **International Journal of Man-Machine Studies 27** (4), 349-370 (October).

Compton, P., Edwards, G., Kang, B., Malor, R., Menzies, T., Preston, P., Srinivasan, A. & Sammut, S. (1991) Ripple down rules: possibilities and limitations. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.?. Banff (October).

Compton, P. & Jansen, R. (1990a) Knowledge in context: a strategy for expert system maintenance. Barter, C.J. & Brooks, M.J., Eds. **AI'88: 2nd Australian Joint Artificial Intelligence Conference, Adelaide Australia, November 1988, Proceedings.** pp.292-306. Berlin: Springer Verlag.

Compton, P. & Jansen, R. (1990b) A philosophical basis for knowledge acquisition. **Knowledge Acquisition 2**(3).241-258.

Gaines, B.R. (1989). An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-Off between Expertise and Data based on Statistically Well-Founded Empirical Induction. **Proceedings of 6th International Workshop on Machine Learning**, pp.156-159. San Mateo, California: Morgan Kaufmann (June).

Gaines, B.R. (1990). An architecture for integrated knowledge acquisition systems. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Fifth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.8-1-8-22. Banff (November).

Gaines, B.R. (1991a). Empirical investigation of knowledge representation servers: design issues and applications experience with KRS. **AAAI Spring Symposium: Implemented Knowledge Representation and Reasoning Systems.** pp. 87-101. Stanford (March) and **SIGART Bulletin** 2(3), 45-56 (June)..

Gaines, B.R. (1991b). Integrating rules in term subsumption knowledge representation servers. **AAAI'91: Proceedings of the Ninth National Conference on Artificial Intelligence.** pp.458-463. Menlo Park, California: AAAI Press.

Gaines, B.R. (1991c). An interactive visual language for term subsumption visual languages. **IJCAI'91: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence.** San Mateo, California: Morgan Kaufmann.

Gaines, B.R. (1991d). The tradeoff between knowledge and data in data acquisition. Piatetsky-Shapiro, G. & Frawley, W. **Knowledge Discovery in Databases.** Cambridge, Massachusetts: MIT Press.

Gaines, B.R. & Shaw, M.L.G. (1990) Cognitive and logical foundations of knowledge acquisition. Boose, J.H. & Gaines, B.R. (Eds) **Proceedings of the Fifth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp. 9-1-9-25. Banff (November).

Horn, P.J., Compton, P.J., Lazarus, L., & Quinlan, J.R. (1985). An expert system for the interpretation of thyroid assays in a clinical laboratory. **Australian Computer Journal, 17,** 7-11.

Li, X. (1991) What's so bad about rule-based programming?. **IEEE Software** 103-105 (September).

Mansuri, Y., Compton, P. & Sammut, C. (1991) A comparison of a manual knowledge acquisition method and an inductive learning method. **Proceedings of the First Knowledge Acquisition for Knowledge-Based Systems Workshop.** pp.114-132. Pokolbin (July).

Michie, D. (1989) Problems of computer-aided concept formation. Quinlan, J.R., Ed. **Applications of Expert Systems Volume 2.** pp.310-333. Sydney: Addison-Wesley.

Nosek, J.T. & Roth, I. (1990) A comparison of formal knowledge representations as communication tools: predicate logic vs semantic network. **International Journal of Man-Machine Studies 33,** 227-239.

Núñez, M. (1990). Decision tree induction using domain knowledge. Wielinga, B., Boose, J.H. & Gaines, B.R., Schreiber, G., van Someren, M., Eds. **Current Trends in Knowledge Acquisition.** pp. 276-288. Amsterdam, IOS.

Piatetsky-Shapiro, G. & Frawley, W. (1991). **Knowledge Discovery in Databases.** Cambridge, Massachusetts: MIT Press.

Quinlan, J.R., Compton, P.J., Horn, K.A. & Lazarus, L. (1987) Inductive knowledge acquisition: a case study. Quinlan, J.R., Ed. **Applications of Expert Systems.** pp.157-173. Sydney: Addison-Wesley.