REFINING INDUCTION INTO KNOWLEDGE

B R Gaines
Knowledge Science Institute, University of Calgary
Calgary, Alberta, Canada T2N 1N4
(gaines@cpsc.ucalgary.ca, (403) 220-5901)

Abstract: Empirical induction systems applied to large databases have the potential to generate knowledge. However, there are many practical problems. First, the database may not contain significant material relevant to induction. It is important to be able to introduce additional knowledge in the inductive process. Second, induced rules do not in themselves constitute humanly comprehensible 'knowledge.' They may be unnecessarily verbose and difficult to understand, and their interactions may be very difficult to comprehend. Some form of post-processing is necessary to make them more comprehensible while maintaining their correctness and completeness. Third, it may be desirable to edit the rules substantially, incorporating additional knowledge, insights and requirements into such post-processing while again controlling correctness and completeness. This paper reports on studies of an inductive tool that combines a term subsumption knowledge representation server with a fast and effective statistical rule induction algorithm in a graphic editing environment to deal with these problems.

INTRODUCTION

As discussed at the 1989 workshop, knowledge discovery from 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.

A knowledge discovery tool 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 another but they will usually be almost impossible for someone to understand without days or weeks of perusal. 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.

This was the starting point for the research reported in this paper, to improve the form of rule generation from empirical induction systems as much as possible automatically, and to give people well-supported means for making further refinements. Along the way other issues have arisen that it has been possible to tackle within the same framework, notably: priming the system with additional knowledge not in the database; inducing rule structures stratified with multi-level exceptions that are more humanly comprehensible; and presenting the resultant knowledge structures in a visual language for ease of understanding and editing.

This paper reports research on a knowledge discovery system that is part of our general toolbox for knowledge acquisition, expertise transfer, knowledge base development, editing, validation and maintenance that commenced with KSS0 (Shaw & Gaines, 1987; Gaines, 1988) and is now KSSn (Gaines 1991a). One problem with research and applications in this area is that of rapid development in techniques, tools and technologies. Systems become obsolescent in a very short time and a monolithic architecture is difficult to maintain and enhance. In the past year KSSn

has been completely re-engineered as a C++ class library implementing a KL-ONE-like knowledge representation server and a set of associated functional modules for knowledge elicitation, text analysis, empirical induction, graphic knowledge base editing, and so on. The opportunity has been taken to incorporate new facilities addressing the issues raised above. In particular the provision of highly interactive graphic editing facilities associated with the general knowledge representation engine has enabled the priming, presentation and editing issues to be addressed. The basic empirical induction is still Induct (Gaines, 1989, 1991b), as reported at the last workshop, but further speeded through a highly optimized re-implementation.

DESIGN OBJECTIVES

The design objectives for the knowledge discovery functionality of KSSn are illustrated in Figure 1. 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 knowledge discovery 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.

The stereotypical caricature of an induction tool consists of the four shaded boxes in Figure 1:

Database→Induction→

Evaluation→Application

The additional interpolated boxes have been added to provide additional functionality to cope with some of the problems noted in the previous section.

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 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,

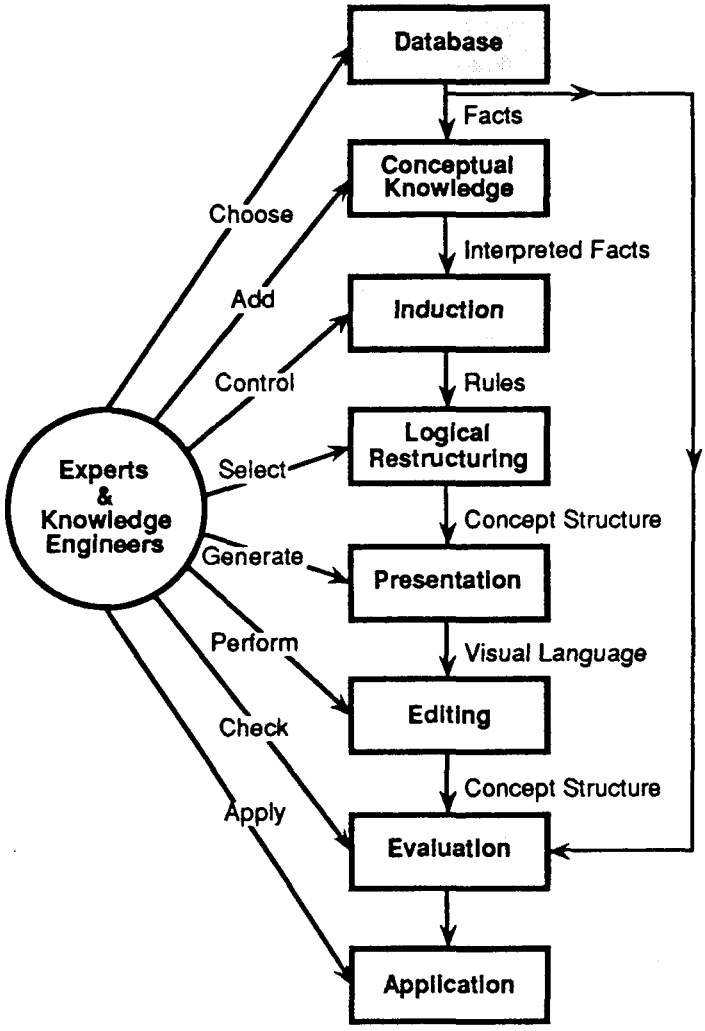

Fig.1 Knowledge discovery process

Lazarus, 1987). If it is not appropriate, given enough data, the induction algorithm will generate a better 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 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, 1991d). 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. Restricted natural language is also inter-translatable (Gomez & Segami, 1990) and provides an alternative to visual languages which we have not yet explored.

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.

SYSTEM ARCHITECTURE

Figure 2 shows the architecture of a knowledge discovery system developed to support the processes of Figure 1. 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, 1991d). 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, 1991c). 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 automatically factored to minimize the knowledge structure.

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 1 can be carried out within the architecture of Figure 2. As noted in the introduction, the components of Figure 2 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.

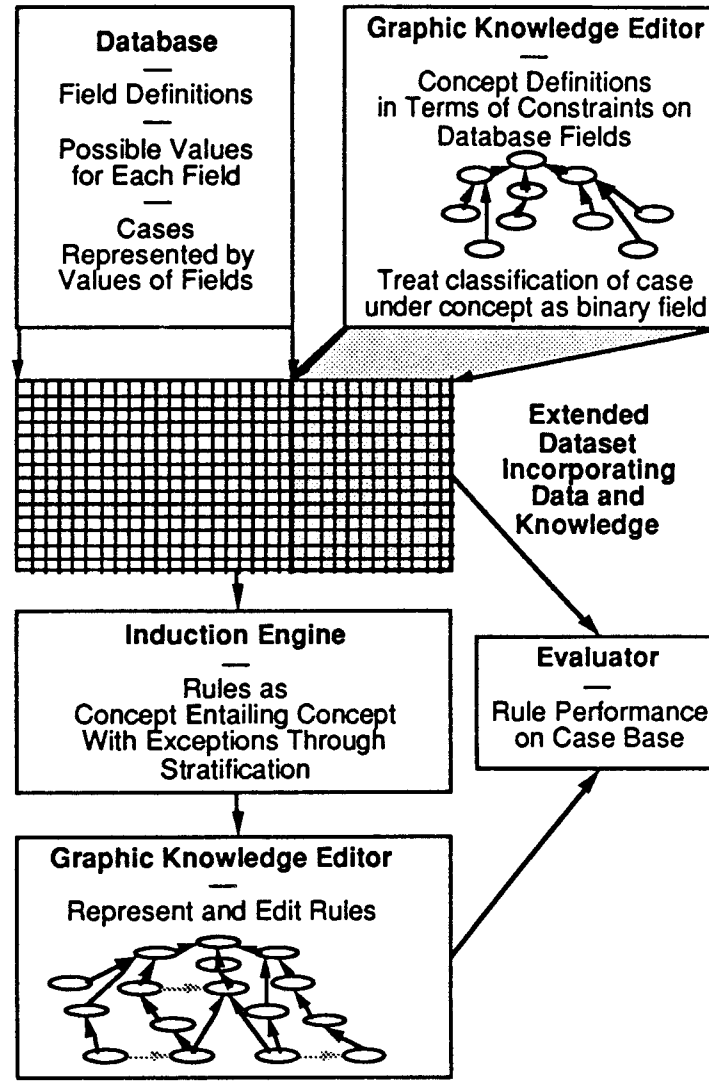

Figure 2 Knowledge discovery system architecture

AN EXAMPLE OF THE SYSTEM IN USE

The best way to illustrate the operation of the system is through a simple example. Figure 3 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 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.

Figure 4 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.

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	-	mm	negative	tail	-	yes	not auto
6	yes	-	-	-	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

Fig.3 Exemplary dataset for shuttle autolander decision (from Michie, 1989)

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*

Fig.4 Decision tree for dataset converted to rules without pruning

: *use auto*
Except:—
vis = yes & errors = mm & sign = negative: *not auto*
vis = yes & stab = no: *not auto*
vis = yes & errors = lx: *not auto*
vis = yes & errors = xl: *not auto*
vis = yes & mag = out: *not auto*
vis = yes & errors = mm & mag = strong & wind = head: *not auto*

Fig.5 Minimal rule set as one default rule and six exceptions

Figure 5 shows a minimal rule set, equivalence in performance to that of Figure 4 generated by Induct from the data of Figure 3. One would expect the smaller rule set in itself to be easier to understand. However, even these rules have scope for further simplification through factorization and restructuring. The "vis=yes" term is common to all of the exceptions and can be factored as a precursor to all of them. The "errors=mm" term is common to two exceptions and could be factored for these. What factorizations are appropriate is not so much a logical matter as a psychological one—how do the experts think about the problem? Representation of the subsumption involved is also most natural in graphical form. The lower part of Figure 6 shows the rules of Figure 5 translated directly to the visual language of the graphic knowledge editor in KSSn with the "vis=yes" term factored out as a separate concept. In the upper part of the figure the database fields and possible values are shown as a domain description.

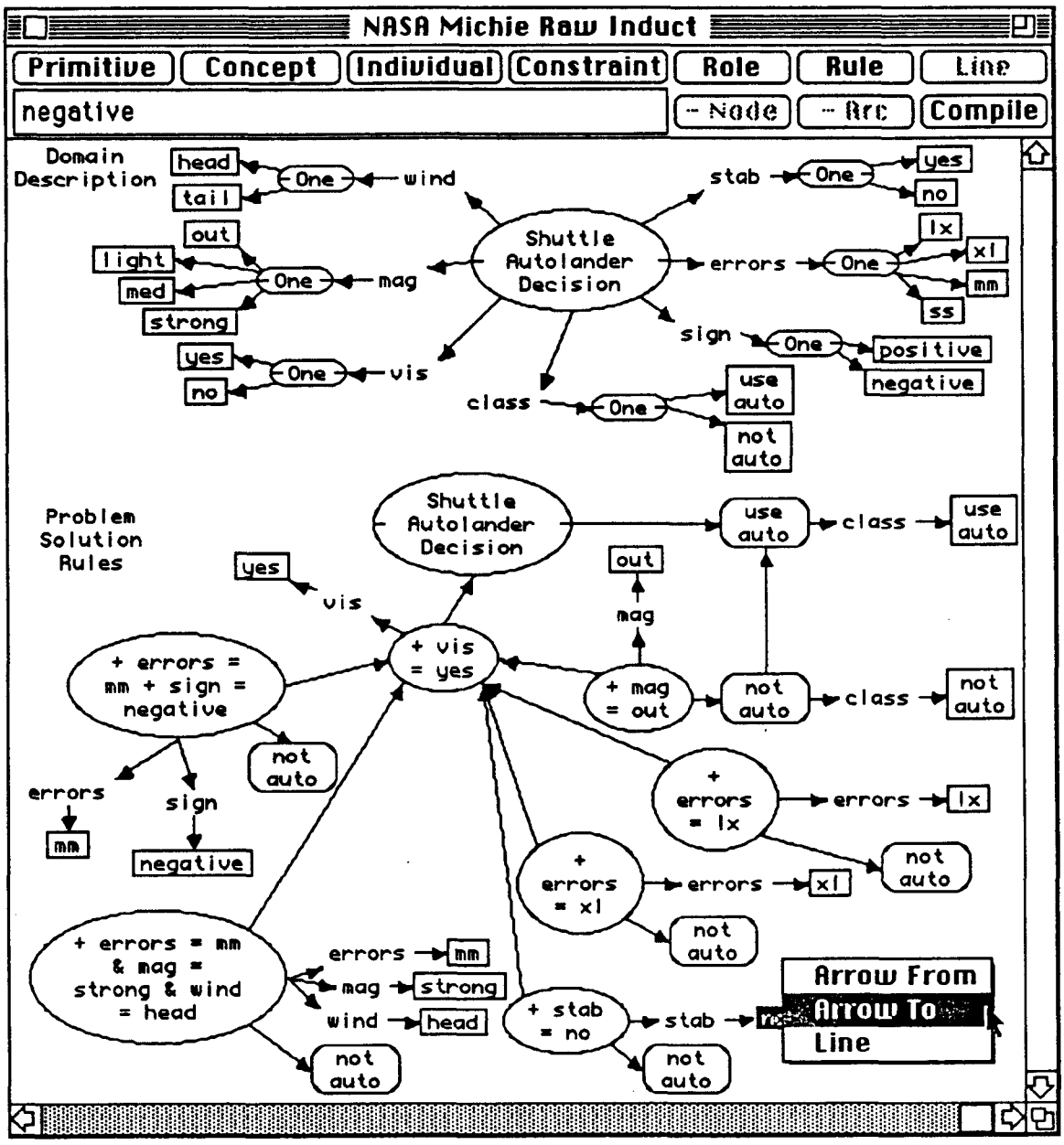

Fig.6 Database domain structure and rules in graphic knowledge editor

The visual language used in Figure 6 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 extensional, 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 6 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.

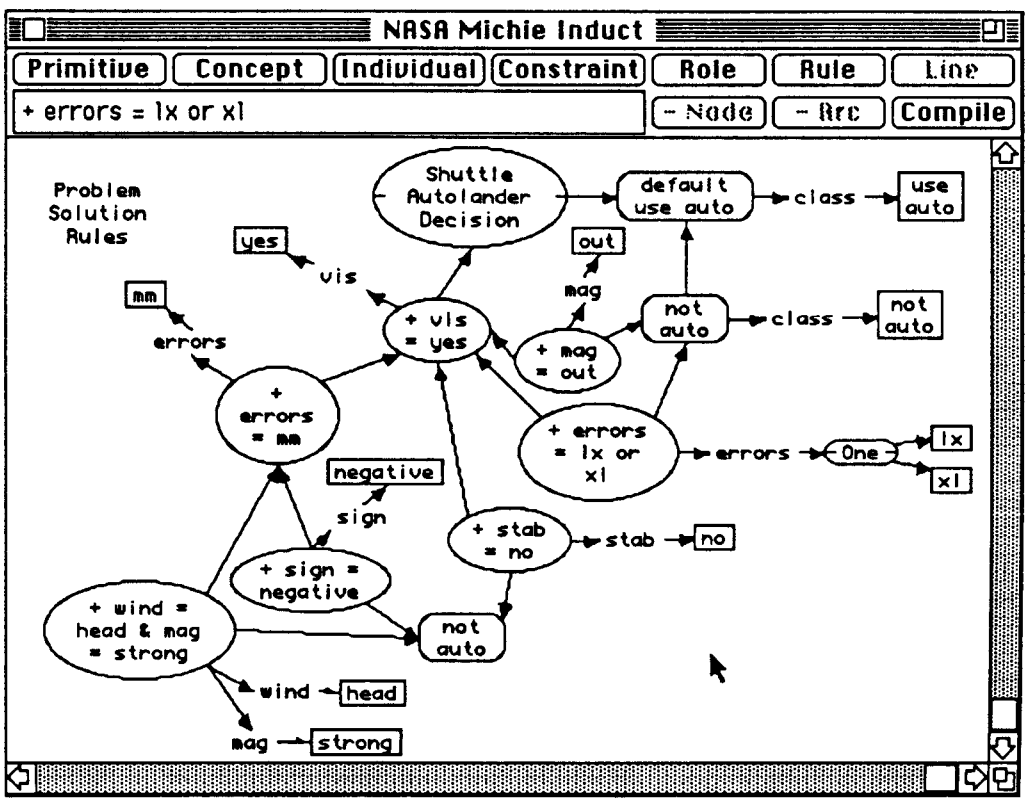

Fig.7 Rules after editing

Figure 7 shows the rules after further editing to factor out "errors=mm" and to express "errors = lx or xl" as a single clause. The upper part of Figure 8 shows a different set of rules developed by taking the default as "not auto" and the bottom part two sample cases entered as queries in the visual language. The conceptual structures expressed in Figures 4 through 8 are equivalent solutions to the original problem in that they all have identical performance. However, they reflect varying degrees of comprehensibility and widely differing knowledge perspectives.

The user can evaluate the knowledge structure against specific cases visually as shown in Figure 8 or through 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.

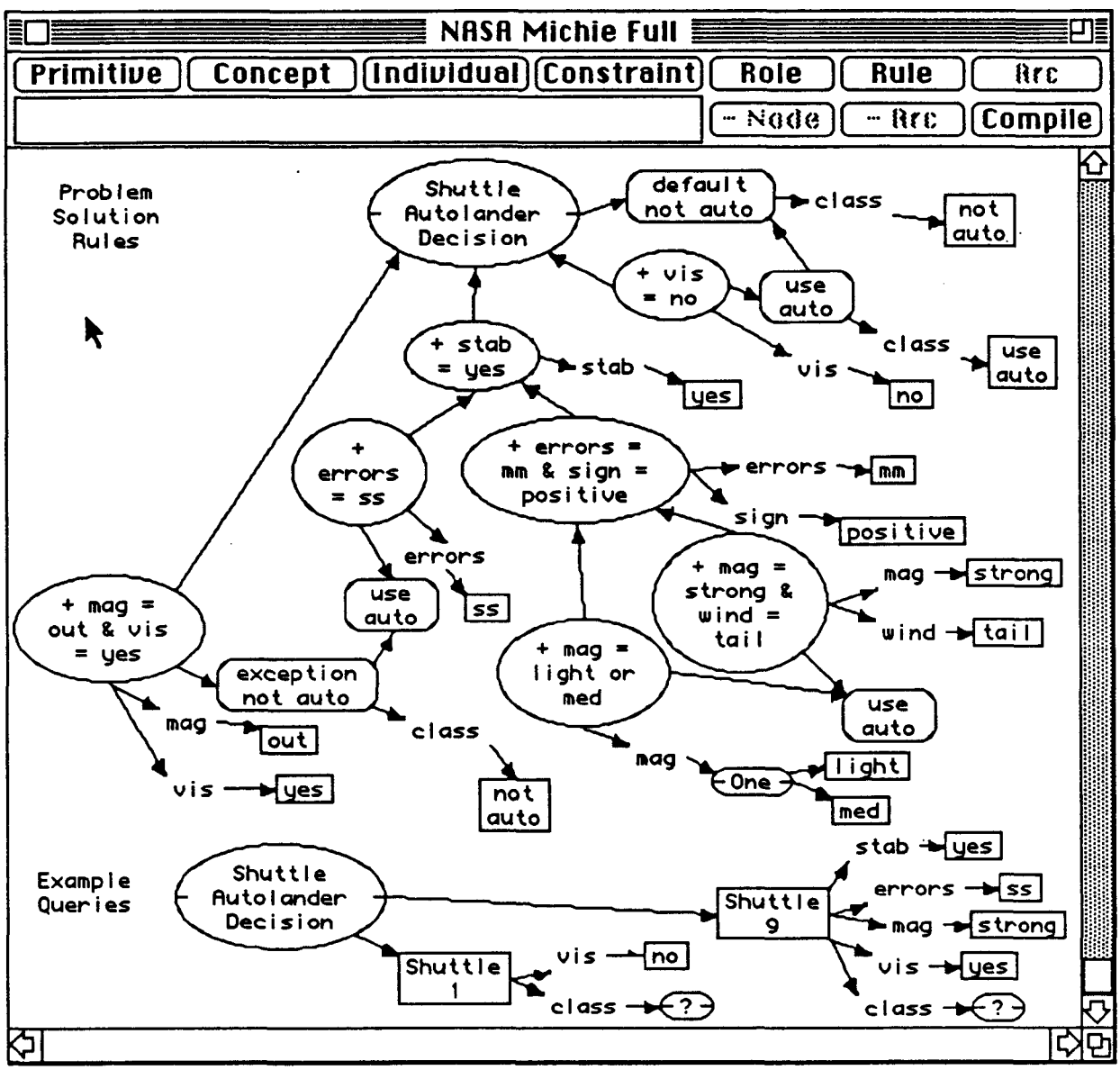

Fig.8 Alternative rules and example queries

KNOWLEDGE REPRESENTATION ISSUES

Some interesting knowledge representation issues have arison in the development of the system described in this paper. The visual representation of knowledge structures for decision making as a hierarchy of concepts and attached rules appears close to the classical treatment in artificial intelligence of default reasoning through semantic networks and inheritance. However, it should first be noted that the rationale for these structures in the current context is not default reasoning but rather the concise and comprehensible presentation of knowledge structures. Second, the usual development based on concept subsumption alone is inadequate. One needs an explicit exemption relation between rules that is independent of the subsumption relation between their premises. Third, because of this, subtle complications can arise.

The knowledge structure in Figure 8 serves to illustrate these issues. It represents the subsumption relations between the left hand sides of the rules explicitly so that exceptions tend to occur lower in the subsumption lattice. However, the exceptions are not themselves generated through subsumption. For example the rule on the left, to not use the autolander when the "mag=out" and the "vis=yes", is an exception to the rule to use it when the "stab=yes" and the "errors=ss", but is also a generally applicable rule not subsumed by the one to which is an exception. If the arrow of subsumption from the concept "+mag=out & vis=yes" went to "+errors=ss" rather than the top level, no decision would be reached about case 6 in Figure 3. The "exception not auto" rule is an exception to the situation in which the "stab=yes" and the "errors=ss" but it is also a directly applicable rule.

Even when subsumption alone is adequate to capture exceptions, related, but more subtle, complications arise when the rule sets are used to generate queries in an inference engine. The over-specialization of a rule can lead to the system requesting more information than is necessary to come to a conclusion. Essentially one needs to add the requirement that, not only is the knowledge structure correct for fully-defined cases, but it is also correct for partially defined cases having a minimal set of attributes specified necessary to arrive at a conclusion.

This section illustrates the nature of exceptions in real datasets, that they interact with the subsumption lattice and fit in with it topologically (as one would expect from logical analysis), but are not dependent upon it. The general exception relation is a partial order imposing stratification on rules.

CONCLUSIONS

Research on, and experience with, knowledge discovery system architectures has been described which has the objective of supplementing empirical induction methodologies with facilities for the use of pre-existing knowledge, the refinement of the rules produced to be humanly meaningful knowledge structures, and support for editing and re-evaluating such structures.

In conclusion, it is suggested that human interaction with the process of knowledge discovery has become an important research issue now that we have powerful induction engines capable of analyzing large databases in a reasonable time. The widespread application of such engines is dependent on providing effective interfaces and additional functionality allowing domain experts to use them in a simple and natural way. We need to be able to combine expertise transfer and empirical induction methodologies to provide powerful tools for real-world applications.

ACKNOWLEDGEMENTS

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada. I am grateful to Paul Compton, Bob Jansen and Ross Quinlan for access to their research and datasets, and to Mildred Shaw for collaborative research and critical comments. I am also grateful to the anononymous referees for some perceptive comments that have improved the paper.

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

Gaines, B.R. (1988) Knowledge acquisition systems for rapid prototyping of expert systems. **INFOR, 26**(4), 256-285 (November).

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. (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)—revized version to appear in SIGART special issue on implemented KR.

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

Gaines, B.R. (1991d). 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. & 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).

Gomez, F. & Segami, C. (1990). Knowledge acquisition from natural language for expert systems based on classification problem-solving methods. **Knowledge Acquisition, 2**(2), 107-128 (June).

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.

Shaw, M.L.G. & Gaines, B.R. (1987). KITTEN: Knowledge Initiation & Transfer Tools for Experts & Novices. **International Journal of Man-Machine Studies, 27**, 251-280.