EXCEPTION DAGS AS KNOWLEDGE STRUCTURES

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

Abstract: The problem of transforming the knowledge bases of performance systems using induced rules or decision trees into comprehensible knowledge structures is addressed. A knowledge structure is developed that generalizes and subsumes production rules, decision trees, and rules with exceptions. It gives rise to a natural complexity measure that allows them to be understood, analyzed and compared on a uniform basis. This structure is a rooted directed acyclic graph with the semantics that nodes are concepts, some of which have attached conclusions, and the arcs are 'isa' inheritance links with disjunctive multiple inheritance. A detailed example is given of the generation of a range of such structures of equivalent performance for a simple problem, and the complexity measure of a particular structure is shown to relate to its perceived complexity. The simplest structures are generated by an algorithm that factors common concepts from the premises of rules. A more complex example of a chess dataset is used to show the value of this technique in generating comprehensible knowledge structures.

INTRODUCTION

This paper presents the continuation of research reported at previous KDD workshops on the development of knowledge structures through the inductive modeling of databases (Gaines, 1991c; Gaines, 1991b). The issue addressed is that of the comprehensibility of the model to people, that is, whether the contents of a model that performs well can be transformed into a communicable knowledge structure that provides insights into the basis of the performance.

The Induct methodology (Gaines, 1989) for deriving rules directly from a database through statistical analysis has proven to be effective in modeling large databases such as the 47,000 cases of the Garvan thyroid medical records over ten years (Gaines and Compton, 1993). The simplicity of the computations underlying Induct enables it to operate at very high speed, on the one hand supporting interactive development of rules, and on the other enabling longitudinal data analysis studies to be undertaken such as the incremental modeling of the Garvan data over a ten year period to determine its stationarity. The performance of the derived models compares favorably with those developed by human experts and by other empirical induction methodologies.

However, the models produced of significant databases typically have such large numbers of rules that they are not comprehensible to people as meaningful knowledge from which they can gain insights into the basis of decision making. This problem is common to both the manually and inductively derived rule sets, and seems intrinsic to the use of production rules as the basis of performance systems (Li, 1991). Similar problems occur with equivalent decision trees where the basis of decision making is not apparent when there are large numbers of nodes.

It is not obvious that an excellent performance system necessarily implies the existence of a comprehensible knowledge structure to be discovered. Human practical reasoning is in major part not knowledge-based (Gaines, 1993b), and in the knowledge acquisition community 'expertise transfer' paradigms have been replaced in recent years by 'knowledge modeling' paradigms (Schreiber, Wielinga and Breuker, 1993) that impute the resultant overt knowledge to the modeling process, not to some hypothetical knowledge base within the expert. Clancey, who has played a major role in promoting this paradigm shift (Clancey, 1989; 1993), has done so in part based on his experience in developing overt knowledge structures from MYCIN rules to support GUIDON (Clancey, 1987a) as a teaching system based on MYCIN. In critiquing MYCIN's rules as not being a comprehensible knowledge structure, Clancey remarks:

"Despite the now apparent shortcomings of MYCIN's rule formalism, we must remember that the program was influential because it worked well. The uniformity of representation, so much the cause of the 'missing knowledge' and 'disguised reasoning' described here, was an important asset. With knowledge so easy to encode, it was perhaps the simple parametrization of the problem that made MYCIN a success. The problem could be built and tested quickly at a time when little was known about the structure and methods of human reasoning. ... We can treat the knowledge base as a reservoir of expertise, something never before captured in quite this way, and use it to suggest better representations." (Clancey, 1987b)

The development of excellent performance systems will remain a major practical objective regardless of the comprehensibility of the basis of their performance. However, the challenge of increasing human knowledge through developing understanding that basis remains a significant research issue in its own right. Can one take a complex knowledge base that is difficult to understand and derive from it a better and more copmprehensible representation as Clancey suggests? If so, to what extent can the derivation process be automated?

This paper presents techniques for restructuring production rules and decision trees to generate more comprehensible knowledge structures. In particular, a knowledge structure is presented that generalizes and subsumes production rules, decision trees, and rules with exceptions. It gives rise to natural complexity measures that allow them to be understood, analyzed and compared on a uniform basis. This structure is a rooted directed acyclic graph with the semantics that nodes are concepts, some of which have attached conclusions, and the arcs are 'isa' inheritance links with disjunctive multiple inheritance.

KNOWLEDGE STRUCTURES

In attempting to improve the comprehensibility of knowledge structures it would be beneficial to have an operational and psychologically well-founded measure of comprehensibility. However, there are in general no such measures. This is not to say that one has to fall back on subjective judgment alone. There are general considerations that smaller structures tend to be more comprehensible, coherent structures more meaningful, those using familiar concepts more understandable, and so on. The internal analog weights and connections of neural networks with graded relations of varying significance are at one extreme of incomprehensibility. Compact sets of production rules are better but not very much so if there are no clear relations between premises or obvious bases of interaction between the rules. Taxonomic structures with inheritance relations between concepts, and concept definitions based on meaningful properties, are probably most assimilable by people, and tend to be the basis of the systematization of knowledge in much of the scientific literature. Ultimately, human judgment determines what is knowledge, but it is not a suitable criterion as a starting point for discovery since the most interesting discoveries are the ones that are surprising. The initial human judgment of what becomes accepted as an excellent knowledge structure may be negative. The process of assimilation and acceptance takes time.

One feature of knowledge structures that is highly significant to discovery systems is that unicity, the existence of one optimal or preferred structure, is the exception rather than the rule. There will be many possible structures with equivalent performance as models, and it is inappropriate to attempt achieve unicity by an arbitrary technical criterion such as minimal size on some measure. The relative evaluation of different knowledge structures of equivalent performance involves complex criteria which are likely to vary from case to case. For example, in some situations there may be preferred concepts that are deemed more appropriate in being usual, familiar, theoretically more interpretable or coherent, and so on. A structure based on these concepts may be preferred over a smaller one that uses less acceptable concepts. Thus, a discovery system that is able to present alternatives and accept diverse criteria for discrimination between them may be preferred over one that attempts to achieve unicity through purely combinatorial or statistical criteria. On the other hand, it is a significant research objective in its own right to attempt to discover technical criteria that have a close correspondence to human judgment.

These have been the considerations underlying the research presented in this paper: to generate knowledge structures that have a form similar to those preferred by people; to explore a variety of possible structures and accept as wide a range of possible of external criteria for assessing them automatically or manually; and to develop principled statistical criteria that correspond to such assessments to the extent that this is possible.

KNOWLEDGE STRUCTURES

The starting point for the research has been empirical induction tools that generate production rules or decision trees. One early conclusion was that a hybrid structure that had some of the characteristics of both rules and trees was preferable to either alone. This structure can be viewed either as a generalization of rules allowing exceptions, or as a generalization of trees such that the criteria at a node do not have to be based on a single property or be mutually exclusive, and the trees are generalized to partial orders allowing branches to come together again. These generalizations allows substantially smaller knowledge structures to be generated than either rules or trees alone, and overcomes problems of trees in dealing with disjunctive concepts involving replication of sub-trees.

Figure 1 exemplifies this knowledge structure, termed an *exception directed acyclic graph* (EDAG). It may be read as a set of rules with exceptions or as a generalized decision tree. Its operational interpretation is that one commences with the root node at concept 0, and places conclusion 0 on a provisional conclusion list. Following the arrow down to the left to concept 1, one evaluates the concept and, if it applies, replaces the provisional conclusion from that path with the specified conclusion. Then one follows the arrows down again to concepts 3 and 4 applying the same logic. Multiple arrows may be conceptualized as being evaluated in parallel—it makes no difference to the results if the graph is followed depth-first or breadth first.

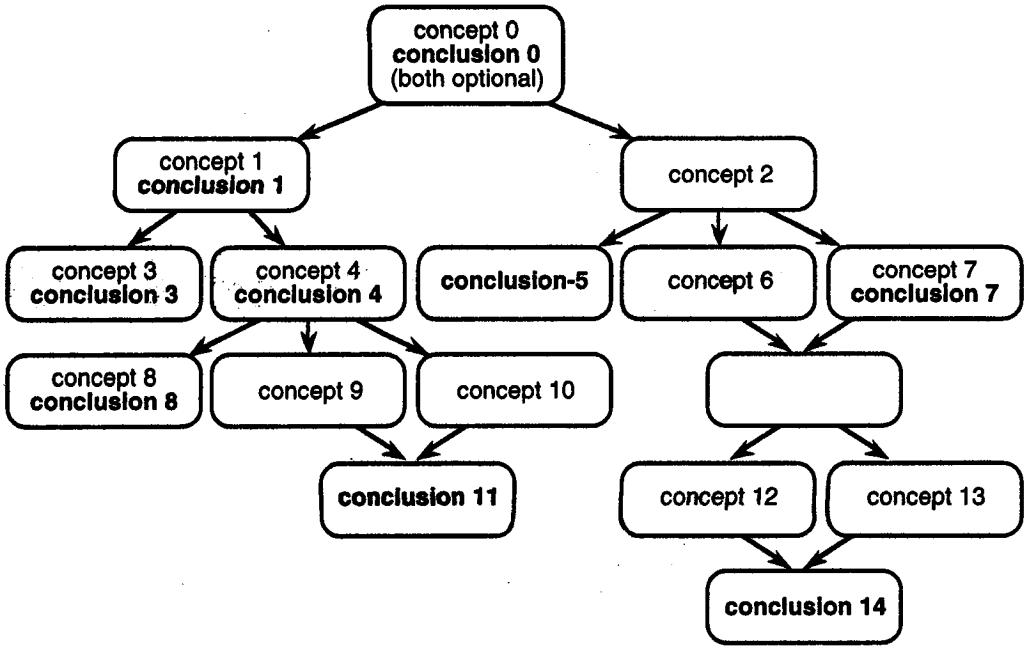

Figure 1 Knowledge structure—exception directed acyclic graph

Several features of EDAGs may be noted:

- Concepts at a branch do not have to be mutually exclusive so multiple conclusions can arise. An interesting example is conclusion 5 which is concluded unconditionally once concepts 0 and 2 apply.
- As shown at concept 4, the structure is not binary. There may be more than two nodes at a branch.
- As shown at conclusion 11, the structure is not a tree. Branches may rejoin, corresponding to rules with a common exception or common conclusion.
- As shown at concept 2, a concept does not have to have a conclusion. It may just be a common factor to all the concepts of its child nodes. As in decision trees, not all concepts in an EDAG are directly premises of rules.
- As shown at conclusion 11, a conclusion does not have to have a concept. It may just be a common conclusion to all the concepts on its parent nodes.

- As shown at the node between concepts 6 and 12, it may be appropriate to insert an empty node to avoid arrows crossing.
- The notion of "concept" used is that of any potentially decidable predicate; that is, one with truth values, true, false or unknown. The unknown case is significant because, as illustrated later, conclusions from the EDAG as a whole may be partially or wholly decidable even though some concepts are undecidable in a particular case—usually because some information is missing for the case.

A set of production rules without a default and without exceptions forms a trivial EDAG with one, empty initial node having an arrow to each rule. An ID3-style decision tree forms an EDAG that is restricted to be a tree, with the set of concepts fanning out from a node restricted to be tests of all the values of a particular attribute. Rules with exceptions form an EDAG in which every node has a conclusion. Rules with exceptions with their premises factored for common concepts (Gaines, 1991b) form a general EDAG.

The direction of arrows in Figure 1 indicates the decision-making paths. However, if the arrows are reversed they become the 'isa' arrows of a semantic network showing inheritance among concepts, with multiple inheritance read disjunctively. For example the complete concept leading to the conclusion 11 is (concept 0 $\wedge$ concept 1 $\wedge$ concept 4) $\wedge$ (concept 9 $\vee$ concept 10).

The complete concept for nodes with child nodes involves the negation of the disjunction of the child nodes, but not of their children. For example, the complete concept leading to the conclusion 1 is (concept 0 $\wedge$ concept 1 $\wedge \neg$(concept 3 $\vee$ concept 4)), with concepts 8 through 10 playing no role.

This last result is important to the comprehensibility of an EDAG. The 'meaning' of a node can be determined in terms of its path(s) back to the root and its child nodes. The other parts of the tree are irrelevant to whether the conclusion at that node will be selected. They remain potentially relevant to the overall conclusion in that they may contribute additional conclusions if the structure is not such that conclusions are mutually exclusive.

From a psychological perspective, the interpretation of the EDAG may be seen as each node providing a well-defined 'context' for its conclusion, where it acts as an exception to nodes back to the root above it, and it has as exceptions the nodes immediately below it. This notion of context is that suggested by Compton and Jansen (1990) in their analysis of 'ripple-down rules', the difference being that EDAG contexts are not necessarily mutually exclusive.

KNOWLEDGE STRUCTURES

Induct is used to generate EDAGs through a three-stage process. First, it is used to generate rules with exceptions. Second, the premises of the rules are factored to extract common sub-concepts (which are not necessarily unique, e.g. A $\wedge$ B, B $\wedge$ C and C $\wedge$ A may be factored in pairs by A, B or C). Third, common exception structures are merged. The second and third stages of factoring can be applied to any set of production rules represented as an EDAG, and it is appropriate to do so to C4.5 and PRISM rules when comparing methodologies and illustrating EDAGs as knowledge structures.

The familiar contact lens prescription problem (Cendrowska, 1987) will be used to illustrate the EDAG as a knowledge structure. Figure 2 shows the ID3 decision tree for the lens problem as an EDAG, and Figure 3 shows the PRISM production rules as an EDAG. The representation in itself adds nothing to the results, but the examples illustrate the subsumption of these two common knowledge structures. The representation of the rules can be improved by factoring the premises to produce the EDAG shown in Figure 4.

Figures 2 through 4 are trivial EDAGs in the sense in that no exceptions are involved. More interesting examples can be generated using C4.5's methodology of: reducing the number of production rules by specifying a default conclusion; removing the rules with that conclusion; and making the other rules exceptions to the default. This can be applied to the PRISM rules in Figure 4 by making "none" the default and removing the four rules with "none" as a conclusion on the left. Both PRISM and C4.5 then generate the same set of rules with exceptions which can be factored into the EDAG shown in Figure 5.

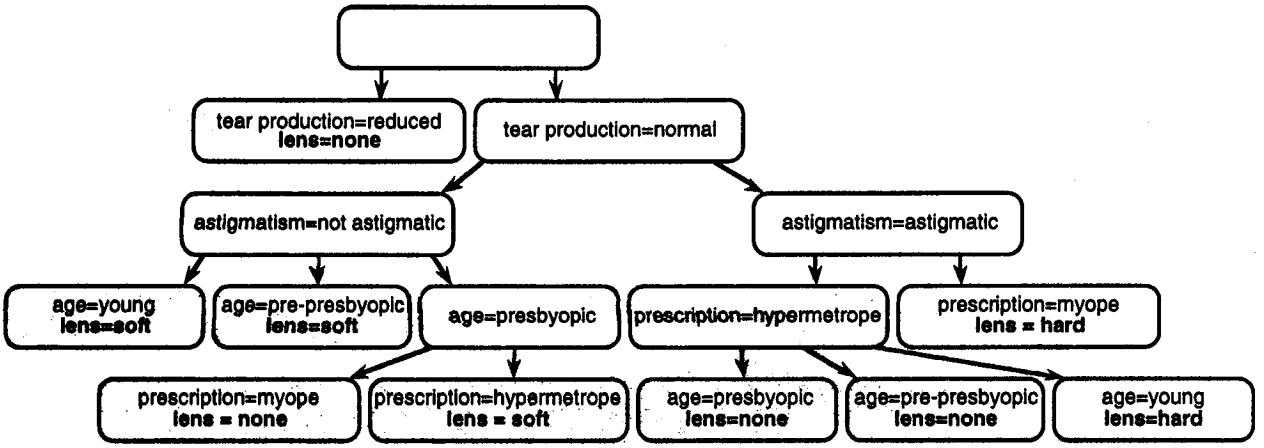

Figure 2 ID3 contact lens decision tree as EDAG

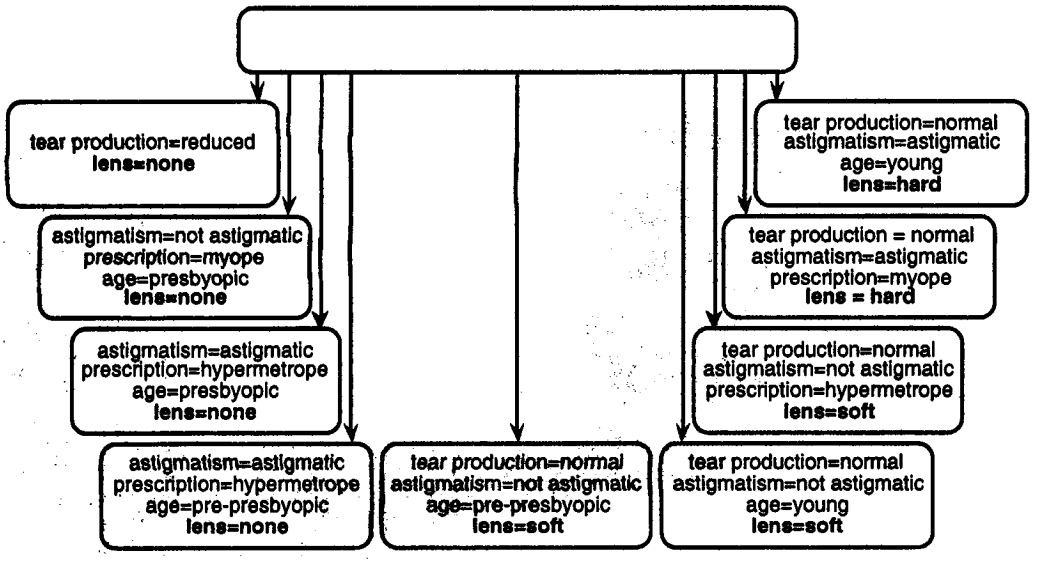

Figure 3 PRISM contact lens production rules as EDAG

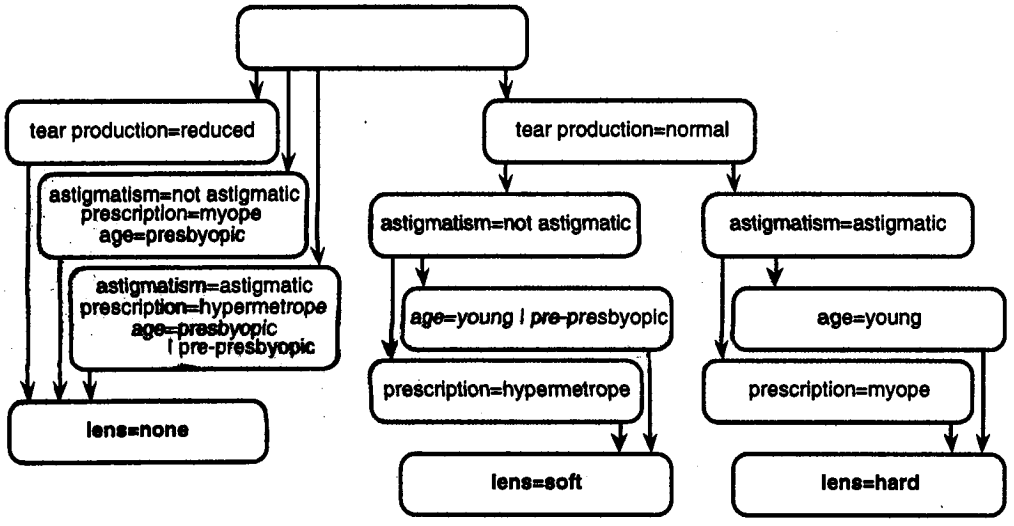

Figure 4 PRISM contact lens production rules, factored as EDAG

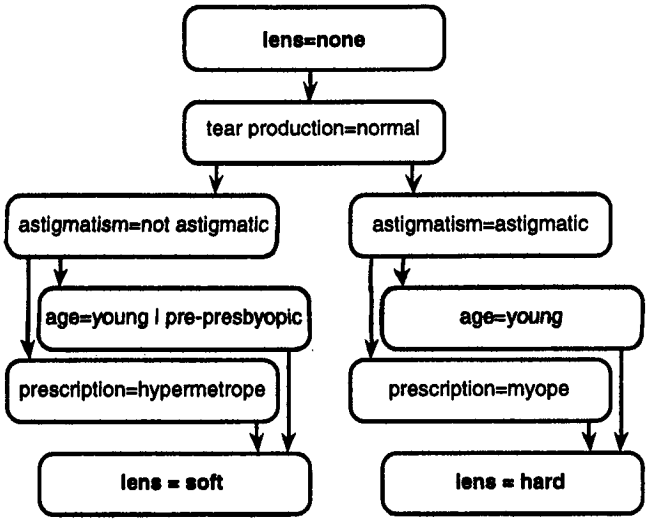

Figure 5 PRISM/C4.5 contact lens rules with default, factored as EDAG

Induct generates multi-level exceptions in that an exception may itself have exceptions. Figure 6 shows two EDAGs generated by Induct from the 24 lens cases. On the left of each EDAG the "soft" conclusion is generated by a simple concept that has an exception. The left EDAG is generated by Induct from the 24 possible cases. The right EDAG is generated by Induct when the data is biased to have a lower proportion of the "hard" exceptions (46 cases used—2*24 with the 2 examples of the "hard" exception removed from the second set—logically unhanged, but the statistical test that Induct uses to determine whether to generate an exception is affected—the exceptions are now statistically more "exceptionable").

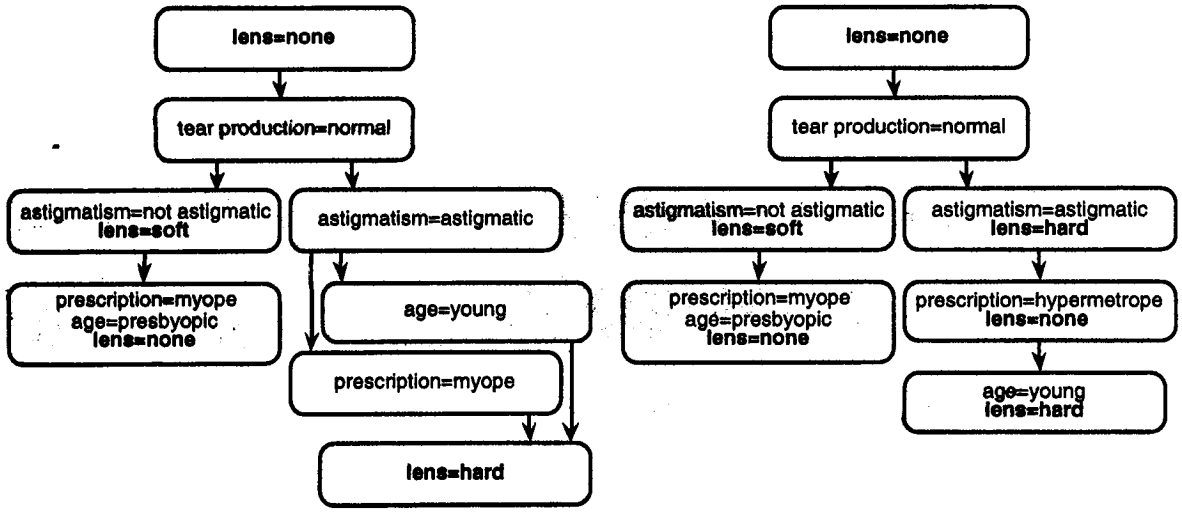

Figure 6 Two forms of Induct rules with exceptions, factored as EDAG

COMPLEXITY MEASURES FOR EDAGs

The results shown in Figures 2 through 6 make it clear that many different EDAGs with the same performance can be derived from existing empirical induction methodologies. It is also apparent that some are substantially simpler than others, and that the simpler one present a more comprehensible knowledge structure. Figures 5 and 6, in particular, illustrate that simple comprehensible knowledge structures are not unique—each represents a reasonable basis for understanding and explaining the contact lens decision methodology. Figure 6 left may appear marginally better than the alternatives in Figures 5 and 6 right, but one can imagine a debate between experts as to which is best—Figure 5 has only one layer of exceptions—Figure 6 right is more balanced in its treatment of astigmatism.

It is interesting to derive a complexity measure for an EDAG that corresponds as closely as possible to the psychological factors leading to judgments of the relative complexity of different EDAGs. As usual, the basis for a structural complexity measure is enumerative, that we count the components in the representation (Gaines, 1977). The obvious components are the node symbols (boxes), the arc symbols (arrows), and the concept and conclusion clauses. In addition, since the graphs are not trees and branches can rejoin with possible line crossings causing visual confusion, the cyclomatic number (Berge, 1958) is considered. It may be regarded as counting (undirected) cycles or as indicating the excess of arcs over those necessary to join nodes. Induct also allows the option of a disjunction of values within a clause, and such a disjunction is weighted by the number of values mentioned, e.g. x=5 counts 1, x∈ {5, 8} counts 2, and x∈ {5, [8, 12]} counts 3 (where [8,12] specifies the interval from 8 through 12).

The derivation of an overall complexity measure from these component counts has to take into account the required graph reductions which should lead to complexity measure reductions. The basic requirements are shown in Figure 7, and it is apparent that clause reduction should dominate, cycle reduction needs to be taken into account with lesser weight, and node reduction with still lesser weight. One suitable formula is:

$$\text{complexity of an EDAG} = (\text{clauses} \times 4 + \text{cycles} \times 2 + \text{nodes} - 1)/5 \qquad (1)$$

where the -1 and scaling by 5 are normalization constants.

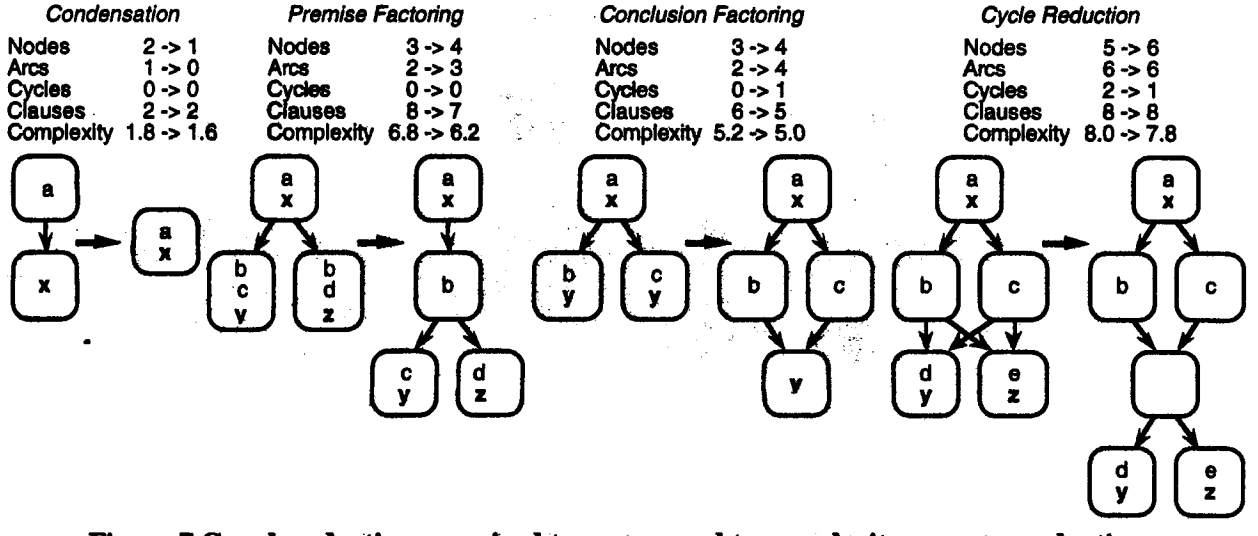

Figure 7 Graph reductions required to correspond to complexity measure reductions

For standard decision trees and production rules there are relations between the numbers in formula (1) that allow one to give the complexity measure more specific interpretations.

$$\text{complexity of decision tree} = \text{number of nodes} + 0.8 \times \text{number of terminal nodes} \qquad (2)$$

$$\text{complexity of production rules} = 0.8 \times \text{number of clauses} + 0.2 \times \text{number of rules} \qquad (3)$$

These both seem intuitively reasonable as reflecting natural components of the structural complexity of trees and rule sets. However, they are somewhat arbitrary, and it would be appropriate to conduct empirical psychological experiments with a range of EDAGs and evaluate the correlation between the understanding of knowledge measured in various ways and the complexity measure developed here. Sufficient data would also allow other complexity measures to be evaluated.

Figure 8 shows the component counts and the computed complexity measures for the solutions to the lens problems shown in Figures 2 through 6. The computed complexity measure does appear to have a reasonable correspondence to the variations in complexity that are subjectively apparent.

Figure	Example	Nodes N	Arcs A	Cycles X=A-N+1	Clauses C	Complexity (4C+2X+N -1)/5
2	ID3	15	14	0	23	21.2
3	PRISM	10	9	0	34	29.0
4	PRISM, factored	14	17	4	19	19.4
5	PRISM/C4.5 default, factored	10	11	2	11	11.4
6L	Induct, factored	8	8	1	11	10.6
6R	Induct variant, factored	7	6	0	13	11.6

Figure 8 Comparison of complexity of contact lens EDAGs

A CHESS EXAMPLE

The contact lens example is useful for illustration but too simple for the significance of the complexity reduction to be evaluated. This section presents a somewhat more complex example based on one of Quinlan's (1979) chess datasets that Cendrowska (1987) also analyzed with Prism. The data consists of 647 cases of a chess end game described in terms of four 3-valued and three 2-valued attributes leading to one of two conclusions. All the solutions described are 100% correct.

Figure 9 shows the 30 node decision tree produced by ID3 and Figure 10 shows the 15 rules produced by C4.5 for the chess data. Cendrowska (1987) reports a substantially larger tree and the same number of rules with some additional clauses.

Figure 11 left and center shows the rules produced by C4.5 factored in 2 ways. The EDAG at the center introduces an additional (empty) node to make it clear that the possible exceptions are the same on both branches. Figure 11 right shows an EDAG produced by Induct. All three EDAGs are interesting in not being trees and involving different simple presentations of the same knowledge.

```
line = 2: safe (324.0)
line = 1:
|   r>>k = 2: safe (162.0)
|   r>>k = 1:
|   |   r>>n = 2: safe (81.0)
|   |   r>>n = 1:
|   |   |   k-n = 1:
|   |   |   |   n-wk = 2: safe (9.0)
|   |   |   |   n-wk = 3: safe (9.0)
|   |   |   |   n-wk = 1:
|   |   |   |   |   k-r = 1: safe (2.0)
|   |   |   |   |   k-r = 2: lost (3.0)
|   |   |   |   |   k-r = 3: lost (3.0)
|   |   |   k-n = 2:
|   |   |   |   n-wk = 2: safe (9.0)
|   |   |   |   n-wk = 3: safe (9.0)
|   |   |   |   n-wk = 1:
|   |   |   |   |   k-r = 2: lost (3.0)
|   |   |   |   |   k-r = 3: lost (3.0)
|   |   |   |   |   k-r = 1:
|   |   |   |   |   |   r-wk = 1: lost (1.0)
|   |   |   |   |   |   r-wk = 2: safe (1.0)
|   |   |   |   |   |   r-wk = 3: safe (1.0)
|   |   |   k-n = 3:
|   |   |   |   k-r = 2: lost (9.0)
|   |   |   |   k-r = 3: lost (9.0)
|   |   |   |   k-r = 1:
|   |   |   |   |   r-wk = 1: lost (3.0)
|   |   |   |   |   r-wk = 2: safe (3.0)
|   |   |   |   |   r-wk = 3: safe (3.0)
```

Figure 9 ID3 chess decision tree

```
                                                    r>>n = 2 -> safe
                                                    r>>k = 2 -> safe
                                                    line = 2 -> safe
                                          k-n = 1 & n-wk = 2 -> safe
                                          k-n = 1 & n-wk = 3 -> safe
                                          k-r = 1 & r-wk = 2 -> safe
                                          k-r = 1 & r-wk = 3 -> safe
                                          k-n = 2 & n-wk = 2 -> safe
                                          k-n = 2 & n-wk = 3 -> safe
k-r = 2 & n-wk = 1 & line = 1 & r>>k = 1 & r>>n = 1 -> lost
k-r = 3 & n-wk = 1 & line = 1 & r>>k = 1 & r>>n = 1 -> lost
n-wk = 1 & r-wk = 1 & line = 1 & r>>k = 1 & r>>n = 1 -> lost
k-n = 3 & r-wk = 1 & line = 1 & r>>k = 1 & r>>n = 1 -> lost
k-n = 3 & k-r = 2 & line = 1 & r>>k = 1 & r>>n = 1 -> lost
k-n = 3 & k-r = 3 & line = 1 & r>>k = 1 & r>>n = 1 -> lost
```

Figure 10 C4.5 chess rules

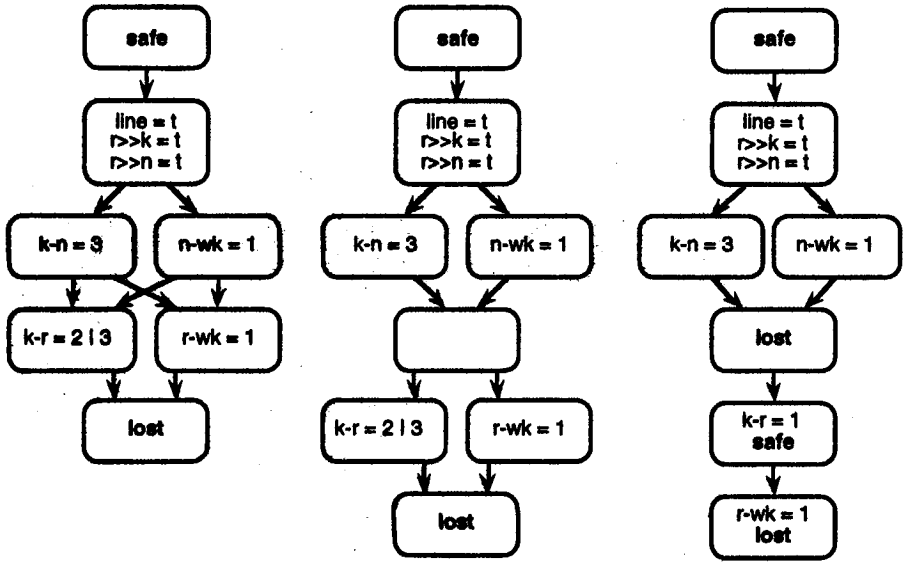

Figure 11 Chess rules with exception, factored as alternative EDAGs

Figure	Example	Nodes N	Arcs A	Cycles X=A-N+1	Clauses C	Complexity (4C+2X+N -1)/5
-	Cendrowska tree	79	78	0	130	119.6
-	PRISM production rules	16	15	0	73	61.4
10	C4.5 production rules	16	15	0	70	59.0
9	ID3 decision tree	31	30	0	50	46.0
-	PRISM default, factored	9	11	3	12	12.4
11L	C4.5 default, factored	7	9	3	10	10.4
11C	C4.5 default+, factored	8	9	2	10	10.2
11R	Induct exceptions, factored	7	7	1	11	10.4

Figure 12 Comparison of complexity of lens EDAGs

Figure 12 shows the complexities of various EDAGs solving the chess problem. The decision tree published by Cendrowska (1987) is more complex than that produced by ID3. PRISM generates the same 15 production rules as does C4.5 except that two of the rules have redundant clauses. This has a small effect on the relative complexity of the production rules, but a larger effect on that of the factored rules since those from PRISM do not factor as well because of the redundant clauses. The three solutions shown in Figure 11 are similar in complexity as one might expect. They are different, yet equally valid, ways of representing the solution.

The reduction between the tree in Figure 9 or the rules in Figure 10 and the EDAGs in Figures 11 does seem to go some way to meeting Michie's objections to trees or rule sets as knowledge structures that Quinlan (1991) cites in the preface of the book from KDD'89. It is more plausible to imagine a chess expert using the decision procedures of Figure 11 than those of Figures 9 or 10.

INFERENCE WITH EDAGs—UNKNOWN VALUES

Inference with EDAGs when all concepts are decidable is simple and has already been described. However, inference when some concepts are unknown as to their truth values involves some subtleties that are significant. Consider the EDAG of Figure 13 where Concept-1 is unknown but Concept-2 is true and the attached conclusion is the same as the default. One can then infer Conclusion = A regardless of what the truth value of Concept-1 might be.

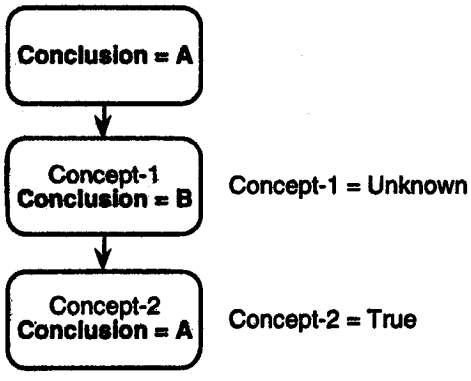

Figure 13 Inference with unknown values

The required reasoning is trivial to implement and is incorporated in the EDAG-inference procedure of KRS (Gaines, 1991a; Gaines, 1993a), a KL-ONE-like knowledge representation server. The evaluation of a concept as true, false or unknown is common in such systems. It is simple to mark the EDAG such that a node is disregarded if: it has an unknown concept but has a child node that is definitely true; or it has the same conclusion as another node whose concept is true.

The importance of pruning the list of possible conclusions is that it is the basis of acquiring further information, for example, by asking the user of the system. It is important not to request information that may be expensive to obtain but is irrelevant to the conclusion. Cendrowska (1987) makes this point strongly in comparing modular rules with decision trees; that, for example, in contact lens prescription the measurement of tear production is time-consuming and unpleasant, and should be avoided if possible. She notes that it is not required in the 3 rules produced by PRISM shown at the lower left of Figure 3 but is the first attribute tested in the ID3 decision tree of Figure 2. She argues that the tree requires the testing of this attribute unnecessarily in certain cases.

However, the three decidable cases with unknown values of tear production (corresponding to the premises of the three rules on the lower left of Figure 3) are all correctly evaluated by the EDAG of Figure 2 using the reasoning defined above, as they are by all the other EDAGs of Figures 3 through 7. The problem Cendrowska raises is a question of properly using the tree as a basis for inference, rather than a distinction between trees and production rules.

Since the complexity figures in Figure 8 indicate that the PRISM rules are, on one reasonable measure, more complex than the ID3 tree, it would seem that the arguments for rules being better than trees are not justified. Certainly the highly restrictive standard decision tree can be improved in comprehensibility by the use of exceptions and factoring, but whether the resultant structure is a more general rule, or a set of rules with exceptions, is a matter of perspective.

The simple pruning procedure described above is sufficient to deal with the tear production problem raised by Cendrowska. However, it is inadequate to properly account for all the nine decidable cases with unknown values (corresponding to the premises of the rules in Figure 3). For example, someone whose tear production is normal, astigmatism is astigmatic and age is young but whose prescription is unknown should be inferred as lens is hard. However, this inference cannot be made with the ID3 tree of Figure 2 using the procedure described above alone. KRS copes with this situation by keeping track of the relation between child nodes that are such that one of them must be true. In the case just defined it infers that the conclusion is lens is hard because the two nodes with lens = hard conclusions at the lower right of Figure 2 are both possible, one of them must be true, and both have the same conclusion.

KRS also disregards a node if the conclusion is already true of the entity being evaluated, again to prevent an unnecessary attempt to acquire further information. The four strategies described are such as to reduce the list of possible conclusions to the logical minimum, and form a complete inference strategy for EDAGs.

It should be noted that the completeness of inference is dependent on the possibility of keeping track of relations between child nodes. This is simple for the classification of single individuals based on attribute-value data. It is far more complex for EDAG-based inference with arbitrary KL-ONE knowledge structures involving related individuals, where it is difficult to keep track of all the relations between partially open inferences. This corresponds to managing the set of possible extensions of the knowledge base generated by resolving the unknown concepts, and is inherently intractable.

CONCLUSIONS

The problem of transforming performance systems based on induced rules or decision trees into comprehensible knowledge structures have been addressed. A knowledge structure, the EDAG, has been developed that generalizes and subsumes production rules, decision trees, and rules with exceptions. It gives rise to a natural complexity measure that allows them to be understood, analyzed and compared on a uniform basis. An EDAG is a rooted directed acyclic graph with the semantics that nodes are concepts, some of which have attached conclusions, and the arcs are 'isa' inheritance links with disjunctive multiple inheritance. A detailed example has been given of the generation of a range of such structures of equivalent performance for a simple problem, and the complexity measure of a particular structure have been shown to relate to its perceived complexity. The simplest structures are generated by an algorithm that factors common concepts from the premises of rules. A second example of a chess dataset was used to show the value of this technique in generating comprehensible knowledge structures.

It is suggested that the techniques described in this paper will be useful in allowing knowledge structures developed by empirical induction that have good performance but are not humanly comprehensible to be transformed into ones that may be more comprehensible.

A referee raised the question of the relative effectiveness of C4.5 and Induct, and it is appropriate to comment briefly on this. There were two motivations underlying the development of Induct. First, to develop a high speed induction engine for use in interactive knowledge acquisition with large datasets. Second, to generate knowledge structures that reflected human conceptual frameworks.

The first objective was addressed by extending the techniques of PRISM to generate rules directly from the data rather than by post-processing of a decision tree, and by using data structures (bit maps) that speeded the computations required. On large datasets Induct is generally over 100 times faster than C4.5 in generating rules with equivalent performance. However, the complexity analysis of the algorithms is similar, and the two could be made comparable in speed by applying the bit map data structures in a C4.5 implementation.

The second objective was addressed by developing rules with exceptions using a statistical methodology to determine whether an over-general rule having error cases is to be specialized by adding additional premise clauses, or left as it is with the exception cases being covered by additional rules. The statistical test used is to calculate whether a rule is successful by chance (Gaines, 1989), and to compare this probability for the rule with errors with those for the rules obtained by adding clauses to it.

What this paper shows is that, for some datasets, the second objective can also be achieved by post-processing the rule sets developed using algorithms such as those of C4.5 to generate EDAGs that are substantially simpler and have a hierarchical structure similar to that of human conceptual frameworks. Experiments are underway to evaluate the post-processing when applied rules derived from data having many decision outcomes where multi-level exceptions seem to generate the most compact and meaningful knowledge structures.

ACKNOWLEDGMENTS

This work was funded in part by the Natural Sciences and Engineering Research Council of Canada. I am grateful to Paul Compton and Ross Quinlan for access to their research and for discussions that have influenced this work. I am also grateful to the anonymous referees for helpful comments.

REFERENCES

Berge, C. (1958). **The Theory of Graphs**. London, Methuen.

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

Clancey, W.J. (1987a). From GUIDON to NEOMYCIN and HERACLES in twenty short lessons. Lamsweerde, A. and Dufour, P., Eds. **Current Issues in Expert Systems**. 79-123. Academic Press.

Clancey, W.J. (1987b). **Knowledge-Based Tutoring: The GUIDON Program**. MIT Press.

Clancey, W.J. (1989). Viewing knowledge bases as qualitative models. **IEEE Expert** 4(2) 9-23.

Clancey, W.J. (1993). The knowledge level reinterpreted: modeling socio-technical systems. **International Journal of Intelligent Systems** 8(1) 33-49.

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

Gaines, B.R. (1977). System identification, approximation and complexity. **International Journal of General Systems** 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 the Sixth International Workshop on Machine Learning**. pp.156-159. Morgan Kaufmann.

Gaines, B.R. (1991a). Empirical investigations of knowledge representation servers: Design issues and applications experience with KRS. **ACM SIGART Bulletin** 2(3) 45-56.

Gaines, B.R. (1991b). Refining induction into knowledge. **Proceedings of the AAAI Workshop on Knowledge Discovery in Databases**. pp.1-10. AAAI.

Gaines, B.R. (1991c). The tradeoff between knowledge and data in data acquisition. Piatetsky-Shapiro, G. and Frawley, W., Ed. **Knowledge Discovery in Databases**. pp.491-505. MIT Press.

Gaines, B.R. (1993a). A class library implementation of a principled open architecture knowledge representation server with plug-in data types. **IJCAI'93: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence**. pp.504-509. Morgan Kaufmann.

Gaines, B.R. (1993b). Modeling practical reasoning. **Int. Journal of Intelligent Systems** 8(1) 51-70.

Gaines, B.R. and Compton, P. (1993). Induction of meta-knowledge about knowledge discovery. **IEEE Transactions on Knowledge and Data Engineering** 5(6) 990-992.

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

Quinlan, J.R. (1979). Discovering rules by induction from large collections of examples. Michie, D., Ed. **Expert Systems in the Micro Electronic Age**. pp.168-201. Edinburgh, Ediinburgh University Press.

Quinlan, J.R. (1991). Foreword. Piatetsky-Shapiro, G. and Frawley, W., Ed. **Knowledge Discovery in Databases**. pp.ix-xii. Cambridge, Massachusetts, MIT Press.

Schreiber, A.Th., Wielinga, B.J. and Breuker, J.A., Ed. (1993). **KADS: A Principled Approach to Knowledge-based System Development**. London, Academic Press.