INDUCING KNOWLEDGE

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

Abstract: One problem with the use of machine learning techniques for knowledge acquisition is that the induced decision structures are rarely humanly comprehensible. What is induced may be a very effective performance system but it does not provide a meaningful model of the basis of that performance. This article addresses the problem of transforming the knowledge bases of performance systems using induced rules or decision trees into comprehensible models. A knowledge structure is developed that generalizes and subsumes production rules, decision trees, and rules with exceptions. It gives rise to complexity measures that allow knowledge structures to be understood, analyzed and compared on a uniform basis. The structure is a directed acyclic graph with the semantics that nodes are premises, some of which have attached conclusions, and the arcs are 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 sub-premises from the premises of rules. More complex examples of chess datasets are used to show the value of this technique in generating comprehensible knowledge structures.

1 INTRODUCTION

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 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, 1993), 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 comprehensible representation as Clancey suggests? If so, to what extent can the derivation process be automated?

This article presents techniques for restructuring production rules and decision trees to generate more comprehensible knowledge structures.

2 COMPREHENSIBILITY OF 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 article: 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.

3 EXCEPTION DIRECTED ACYCLIC GRAPHS (EDAGs)

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 allow substantially smaller knowledge structures to be generated than either rules or trees alone, and overcome 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:

- All paths through the graph are traced from each of its root nodes.
- For each node on a path, if the premise (if any) is true then the conclusion (if any) is noted for that path, replacing any previous conclusion noted for the path.
- A path is traced until a premise fails to hold, or a leaf node is reached.
- When a path has been traced, any conclusion noted is asserted.

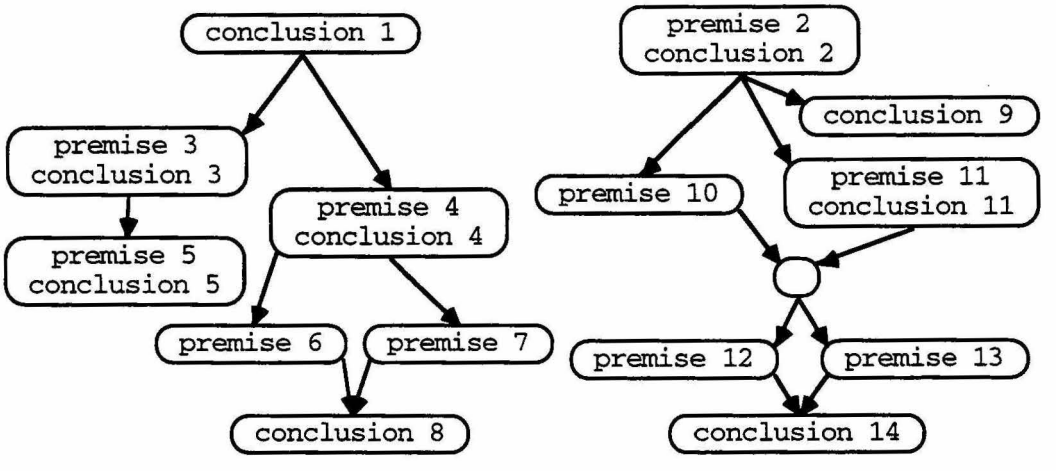


Figure 1 Knowledge structure—exception directed acyclic graph

Several features of EDAGs may be noted:

- The graph is not rooted and may have several disconnected parts.
- Concepts at a branch do not have to be mutually exclusive, so multiple conclusions can arise. An interesting example is conclusion 9 which is concluded unconditionally once premise 2 applies.
- As shown at premise 2, the structure is not binary. There may be more than two nodes at a branch.

- As shown at conclusion 8, the structure is not a tree. Branches may rejoin, corresponding to rules with a common exception or common conclusion.
- As shown at premise 6, a premise does not have to have a conclusion. It may just be a common factor to all the premises of its child nodes. As in decision trees, not all premises in an EDAG are directly premises of rules.
- As shown at conclusions 1 and 8, a conclusion does not have to have a premise. It may be a default conclusion, or a common conclusion to all the premises on its parent nodes.
- As shown by the null node between premises 10 and 12, it may be appropriate to insert an empty node to avoid arrows crossing.
- The notion of "premise" 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 premises 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 no connections. An ID3-style decision tree forms an EDAG that is restricted to be a tree, with the set of premises 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 sub-premises (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 premises, with multiple inheritance read disjunctively. For example the complete premise leading to the conclusion 14 is premise 2 $\wedge$ (premise 10 $\vee$ premise 11) $\wedge$ (premise 12 $\vee$ premise 13).

The complete premise for nodes with child nodes involves the negation of the disjunction of the child nodes, but not of their children. For example, the complete premise leading to the conclusion 1 is $\neg$(premise 3 $\vee$ premise 4), with premises 5 through 7 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.

4 INDUCING EDAGs

Induct (Gaines, 1989) 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-premises (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. The problem involves three 2-valued attributes, one 3-valued attribute, and a 3-valued conclusion. 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.

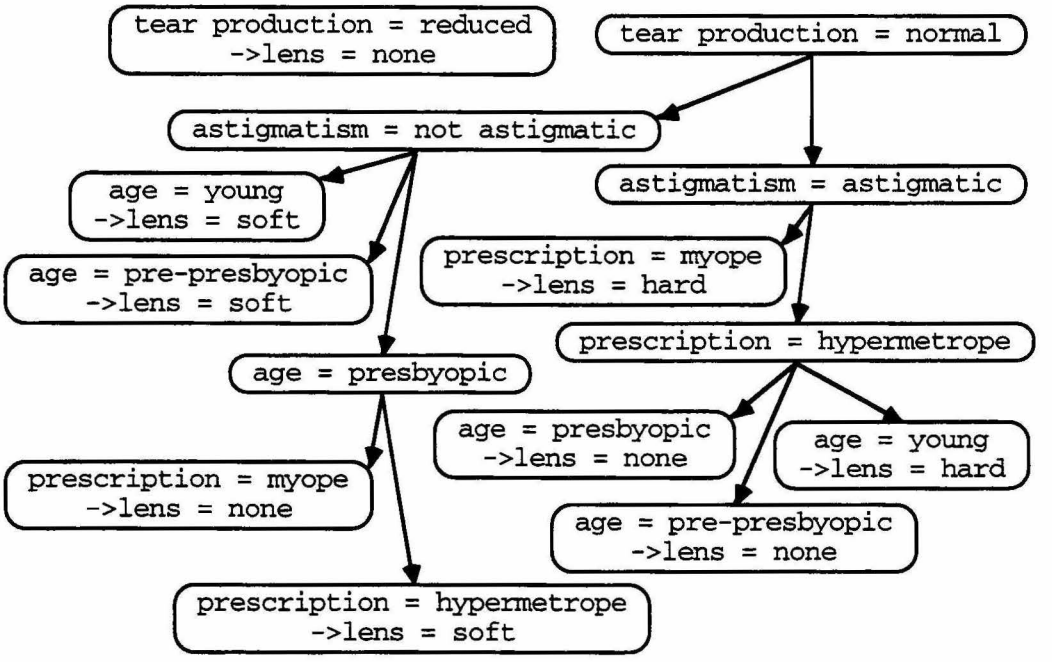


Figure 2 ID3 contact lens decision tree as a simple case of an EDAG

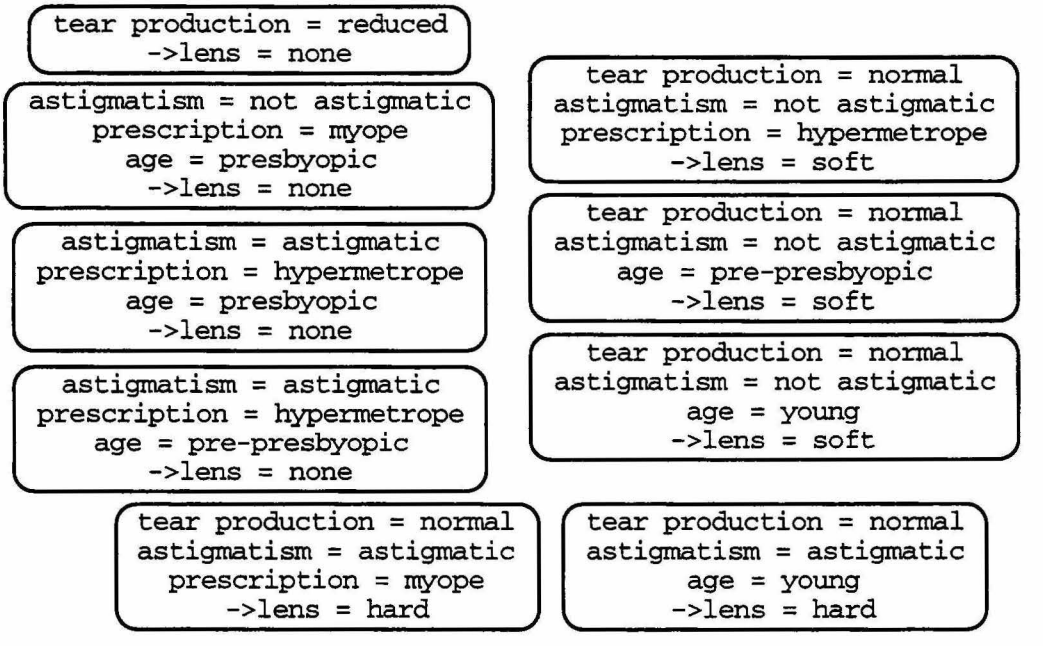


Figure 3 PRISM contact lens production rules as a simple case of an EDAG

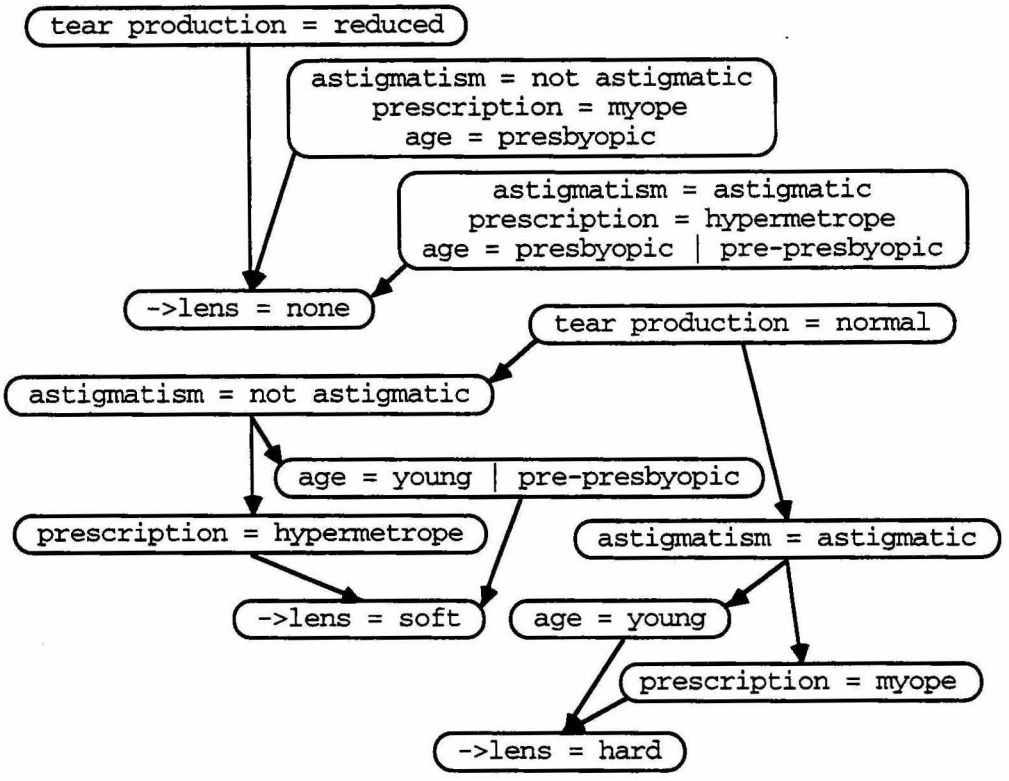


Figure 4 PRISM contact lens production rules, factored as EDAG

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.

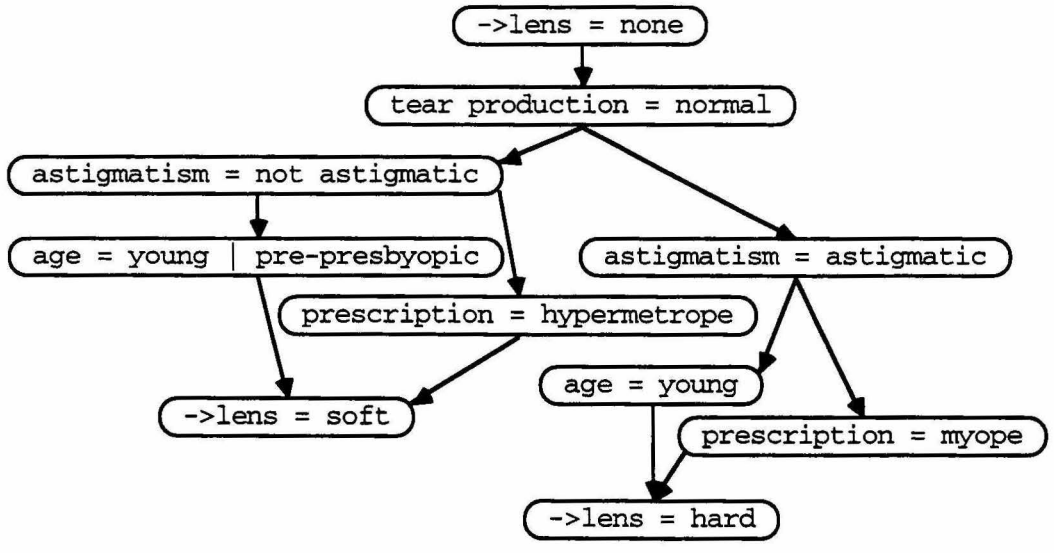


Figure 5 PRISM/C4.5 contact lens rules with default, factored as EDAG

Induct generates multi-level exceptions in that an exception may itself have exceptions. Figure 6 shows the EDAG generated by Induct from the 24 lens cases. On the left, the "soft" conclusion is generated by a simple premise that has an exception. On the right, the "hard" conclusion is by the same rules as in Figure 5.

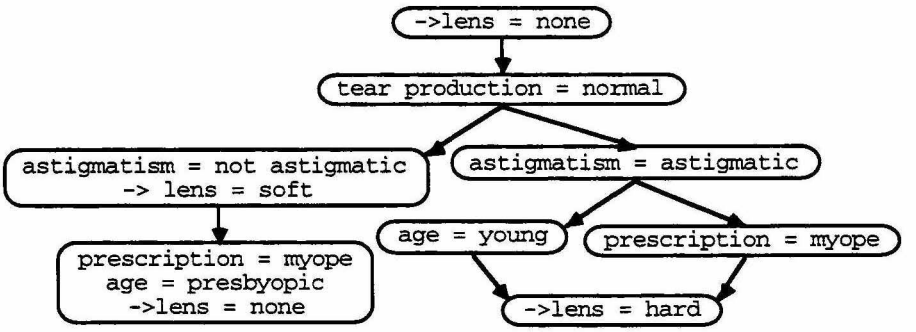


Figure 6 Induct rules with exceptions, factored as EDAG

Varying the distribution of the data set can lead to different exception structures. Figure 7 shows the EDAG is generated by Induct when the data is biased to have a lower proportion of the "hard" exceptions. 46 cases have been used—2 x 24 with the 2 examples of the "hard" exception removed from the second set. The data is logically unchanged, but the statistical test that Induct uses to determine whether to generate an exception is affected. The exceptions are now statistically more "exceptional".

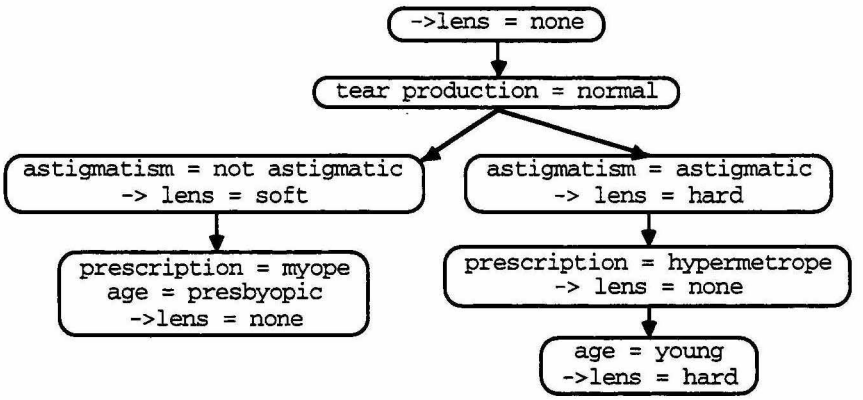


Figure 7 Alternative Induct rules with exceptions, factored as EDAG

5 COMPLEXITY MEASURES FOR EDAGs

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, in that one counts the components in the representation (Gaines, 1977). The obvious components are the nodes, the arcs, and the premise and conclusion clauses. Since the graphs are not trees, branches can rejoin with possible line crossings causing visual confusion. This correlates with the extent to which nodes are disjunctive and have several outgoing arcs. A possible measure is the number of outgoing arcs in excess of one for a node, which may be calculated as arcs + final nodes - 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, $x \in \{5, 8\}$ counts 2, and $x \in \{5, [8, 12]\}$ 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 8. It is apparent that clause and crossing reduction should dominate node reduction.

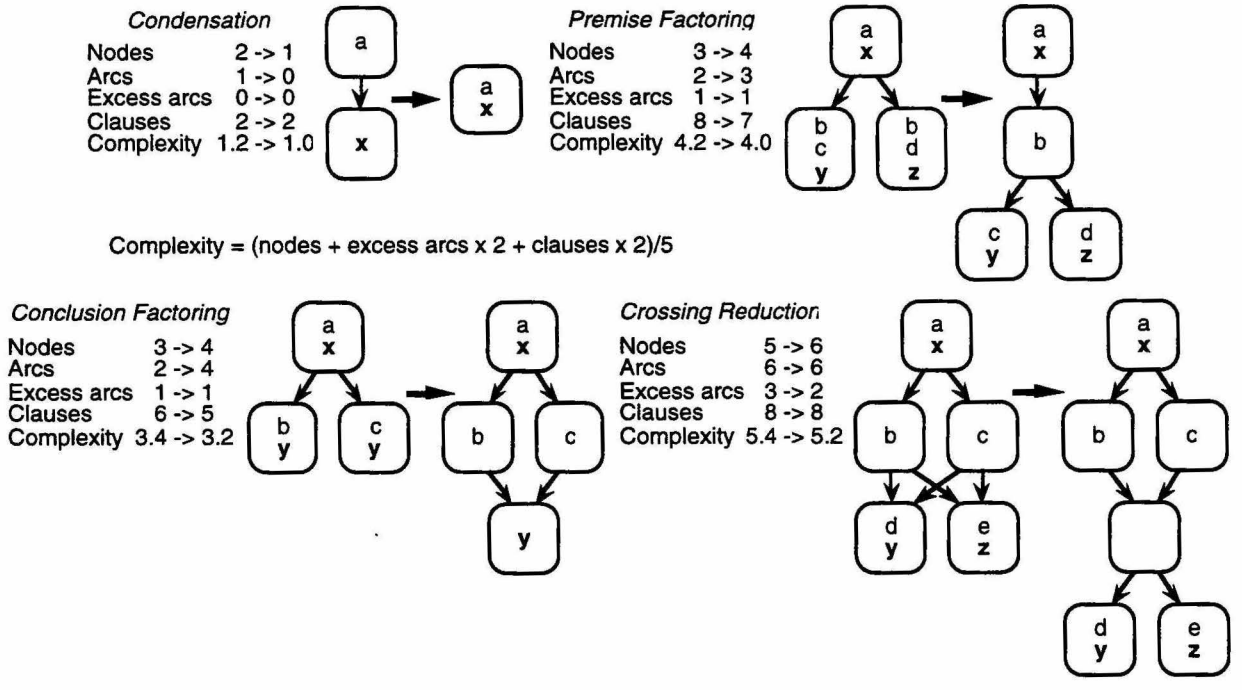


Figure 8 Complexity reductions required to correspond to graph reductions

One suitable formula is that shown in Figure 8, that the:

complexity of an EDAG = (nodes + excess arcs x 2 + clauses x 2)/5 (1)

where the scaling by 5 is for normalization.

For binary 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.:

complexity of a binary decision tree = number of nodes (2)

which is the usual measure of the complexity of a binary decision tree;

complexity of production rules = 0.6 x number of rules + 0.4 x number of premise clauses (3)

which is a weighted mean of two usual measures of the complexity of a set of production rules.

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. Nosek and Roth's (1990) empirical comparison of formal knowledge representation schema exemplifies what can be done. Shum and Hammond (1994) have surveyed studies of comprehensibility of data structures more generally.

Figure 9 shows the component counts and the computed complexity measures for the solutions to the lens problems shown in Figures 2 through 7. The computed complexity measure does appear to have a reasonable correspondence to the variations in complexity that are subjectively apparent.

Fig.	Example	Nodes N	Final Nodes F	Arcs A	Excess E = A +F-N	Clauses C	Complexity (N+2E+2C)/5
2	ID3	14	9	12	7	23	14.8
3	PRISM	9	9	0	0	34	15.4
4	PRISM, factored	13	3	13	3	19	11.4
5	C4.5 default, factored	10	2	11	3	11	7.6
6	Induct, factored	8	2	8	2	11	6.8
7	Induct variant, factored	7	2	6	1	13	7.0

Figure 9 Comparison of complexity of contact lens EDAGs

6 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 rook versus knight end game situation 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 10 shows the 30 node decision tree produced by ID3 for the chess data.

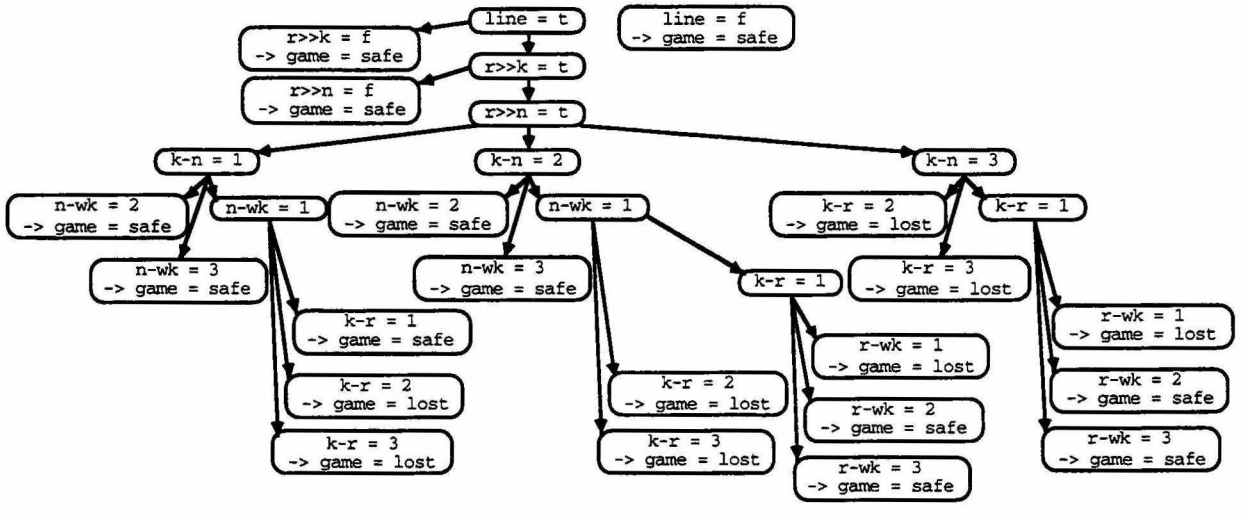


Figure 10 ID3 decision tree solving a rook versus knight chess end game problem
(l ine = t if the black king, rook and knight are in line, r>>k = t if the rook bears on the king, r>>n = t if the rook bears on the knight, k-r is the king to rook distance, and so on)

Figure 11 shows the 15 rules produced by C4.5 from the ID3 tree of Figure 10. Figure 12 shows the rules of Figure 2 with the conclusion "game = safe" taken as a default and common clauses in the premises factored by Induct. The numbers indicate that 613 cases are covered by the default, and 34 are exceptions. The EDAG in Figure 13 is the same as that in Figure 12 with an additional (empty) node introduced to make it clear that the possible exceptions are the same on both branches. Figure 14 shows a solution with multi-level exceptions directly induced by Induct. The numbers indicate that the exceptions may themselves be represented as a default covering 30 cases, an exception to that covering 10 cases, and an exception to that covering 4 cases.

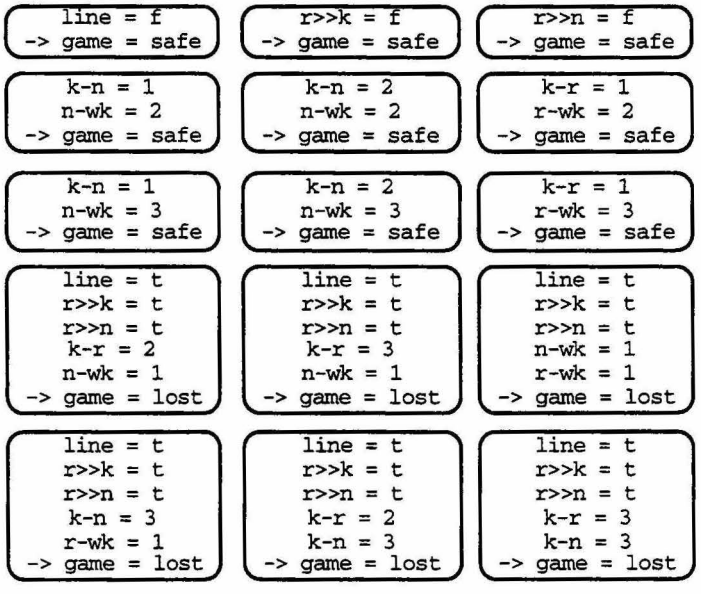


Figure 11 C4.5 rules derived from the decision tree of Figure 10

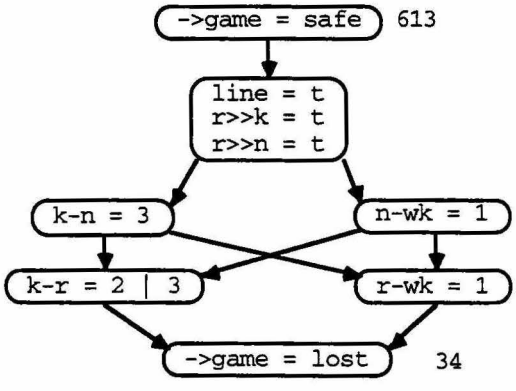


Figure 12 C4.5 chess rules with exception, factored as EDAG

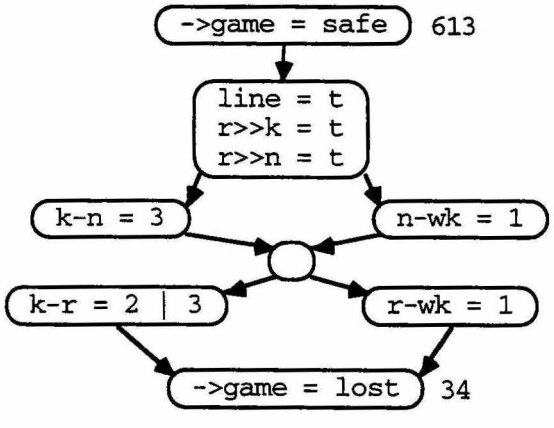


Figure 13 C4.5 chess rules with exception, factored with cycle reduction

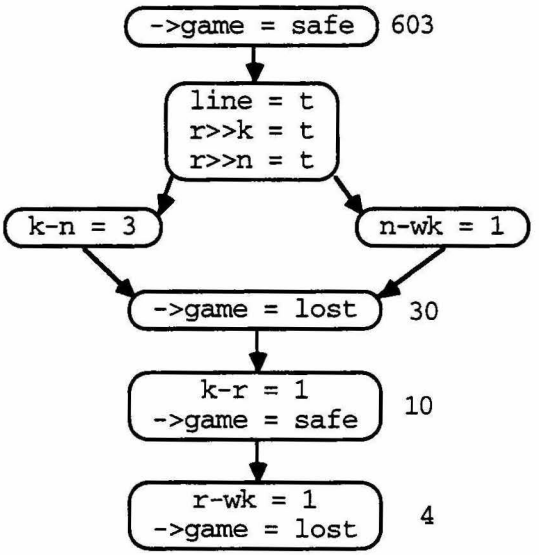


Figure 14 Induct chess rules with exceptions, factored as EDAG

The EDAGs of Figures 12 through 14 are logically equivalent to those of Figures 10 and 11 but substantially smaller and hence easier to understand. It would be very tedious to explain the decision tree or rules in words. However, the small EDAG in Figure 13 may be explicated as "the game is safe unless the black king, rook and knight are in line, the rook bears on the king, the rook bears on the knight, and either the king to knight distance is 3, or the knight to white king distance is 1, and either the king to rook distance is 2 or 3, or the rook to white king distance is 1." The EDAG in Figure 14 may be explicated as "the game is safe unless the black king, rook and knight are in line, the rook bears on the king, the rook bears on the knight, and either the king to knight distance is 3, or the knight to white king distance is 1, when the game is lost, unless the king to rook distance is 1, when it is won, unless the rook to white king distance is 1, when it is lost."

In both cases the graphic representation in the EDAGs seems more perspicuous than the text. EDAGs retain the flow of decision making that makes decision trees attractive. In addition, the reduced restrictions of the directed acyclic graph structure enables them to avoid some of the unnecessarily complex structures generated by redundant replication of nodes in a decision tree representing a disjunctive structure.

Fig.	Example	Nodes N	Final Nodes F	Arcs A	Excess E = A +F-N	Clauses C	Complexity (N+2E+2C)/5
-	Cendrowska tree	78	52	76	50	130	87.6
-	PRISM production rules	15	15	0	0	73	32.2
10	ID3 decision tree	30	20	28	18	50	33.2
11	C4.5 production rules	15	15	0	0	70	31.0
-	PRISM default, factored	9	1	11	3	12	7.8
12	C4.5 default, factored	7	1	9	3	10	6.6
13	C4.5 default+, factored	8	1	9	2	10	6.4
14	Induct exceptions, factored	7	1	7	1	11	6.2

Figure 15 Comparison of complexity of lens EDAGs

Figure 15 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 Figures 12 through 14 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 10 or the rules in Figure 11 and the EDAGs in Figures 12 through 14 does seem to go some way to meeting Michie's objections to trees or rule sets as knowledge structures that Quinlan (1991) cites. It is more plausible to imagine a chess expert using the decision procedures of Figures 12 through 14 than those of Figures 10 or 11.

7 COMPREHENSIBILITY AND SCALABILITY—STRUCTURED INDUCTION

How do the results of the previous sections scale up to a more complex situation? Shapiro (1987) studied a situation in the rook versus pawn (KPa7KR) end-game situation which he notes is poorly documented in the chess literature. The white pawn is at a7, potentially able to queen in one move, and the black rook and the two kings are in any of the 209,718 legal configurations of which 129,825 are classified won-for-white and the remaining 79,893 are classified not-won-for-white.

In the chess situations studied by Quinlan and Shapiro the board positions of the pieces are well-defined but the attributes that a chess expert uses in assessing those positions are not. Hence, human experts were interviewed to elicit those characteristics of the situations that they saw as significant to making a decision about whether the game was won or not. Quinlan's studies elicited these attributes, applied them to a set of situations where the outcome was known, and then used ID3 to develop a decision tree which would be totally correct if the attributes adequately characterized the problem.

Shapiro's study addressed the problem noted in the initial quotation, that the resultant decision trees, although correct, could not be regarded as demonstrating "knowledge" of chess. He had chess experts partition the problem situation and define small sub-problems, each of which defined a new decision variable. He defined "small" to mean that the sub-problem could be solved using fewer than 7 attributes, each of which could be derived from a board position with less than a "screenfull of C code." For each sub-problem, he classified the database of cases in terms of the small set of attributes and the decision variable, and then used ID3 to develop a decision tree.

Shapiro termed this technique "structured induction" and used it to develop a complete solution of the KPa7KR end-game in terms of 9 sub-problems introducing 8 additional decision variables. The total number of attributes defined is 35 binary and 1 ternary, and the total number of different attribute vectors involved is 3,196. This dataset is available through the University of Irvine archive (Murphy and Aha, 1994), and has been used in other studies (Muggleton, 1987; Holte, Acker and Porter, 1989).

7.1 Structured Induction using Decision Trees

Shapiro's use of decision trees reflects the tools and representational schema available to him in the early 1980s. Figure 16 is reproduced from Shapiro's book and shows the structure of his solution in terms of the attributes used to develop the 9 sub-trees. Each circle represents a decision tree with between 4 and 7 attributes, and the complete structure has 122 nodes.

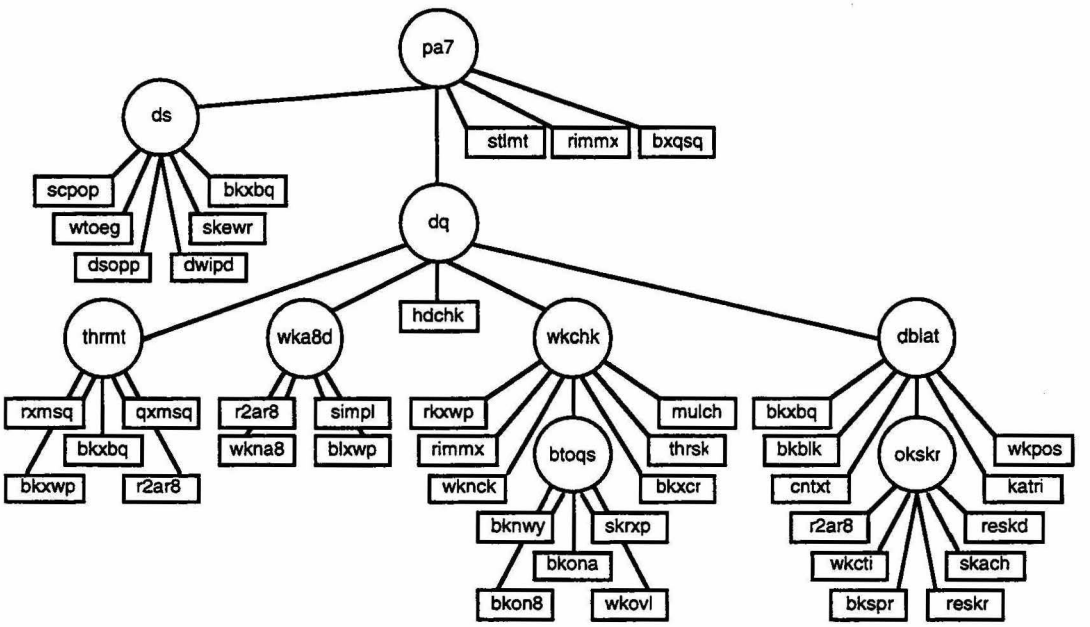


Figure 16 Structured induction of solution to a chess end game (Shapiro, 1987)

Figure 17 shows the top-level decision tree, pa7 in Figure 16. It involves the introduction of intermediate decision variables, such as ds (good delayed skewer threat) and dq (simple delay to queening), which are computed through other decision trees.

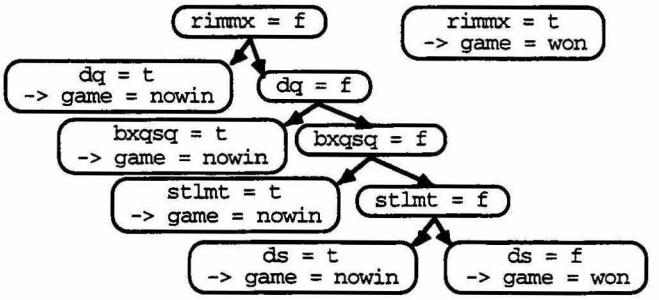


Figure 17 Top-level tree pa7 (Shapiro, 1987)

Figure 18 shows the decision tree for computing ds.

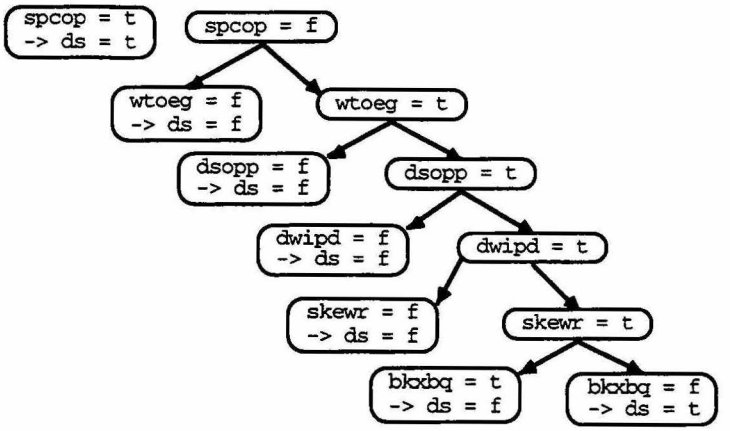


Figure 18 Sub-tree to determine ds (Shapiro, 1987)

Figure 19 shows the complete structured induction solution developed as a single EDAG by combining the trees to eliminate the intermediate decision variables. Context boxes have been placed around each sub-tree to retain the partition into sub-problems.

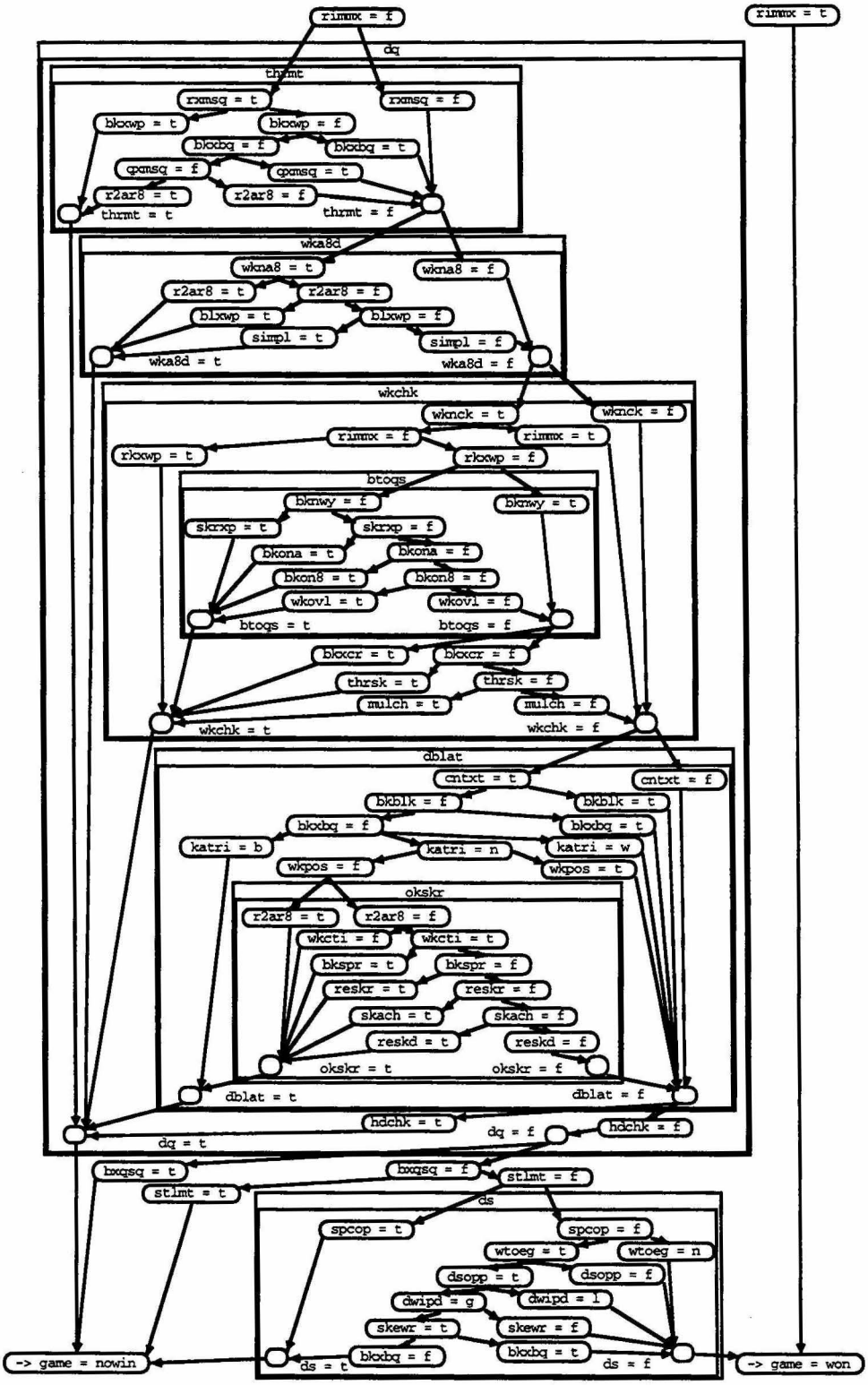


Figure 19 Structured induction trees combined by elimination of intermediate variables

7.2 Reconstruction of Structured Induction using EDAGs

As a first experiment ID3 was replaced with Induct and an EDAG induced for each sub-problem. Figure 20 shows the 9 EDAGs induced for the sub-problems. Enclosures have been added to show the sub-problems and their relationships. A nested EDAG computes an intermediate decision variable used by the EDAG at the level of nesting above.

It can be seen at the bottom right that the game is won unless the black rook cannot be captured safely (rimmx = f) and either there is a simple delay to queening (dq = t) or one or more black pieces controls the queening square (bxqsq = t), or the white king is in stalemate (stlmt = t), or there is a good delayed skewer threat (ds = t).

It can be seen at the bottom left that there is a good delayed skewer threat (ds = t) if there is a special opposition pattern (spcop = t), or a combination of 5 other attributes have the values specified. Each of the 8 intermediate decision variables is determined by a relatively simple and comprehensible EDAG. Those for wka8d (white king on promotion square) and wkchk (white king in check) are interesting because Induct has generated two-level exceptions for them.

Thus, the reconstruction of Shapiro's results in terms of EDAG's confirms more clearly than did the original ID3 trees that the problem has been structured into simple and sub-problems. The sub-problems are also meaningful because they correspond to situations defined by the expert in terms comprehensible to a chess player.

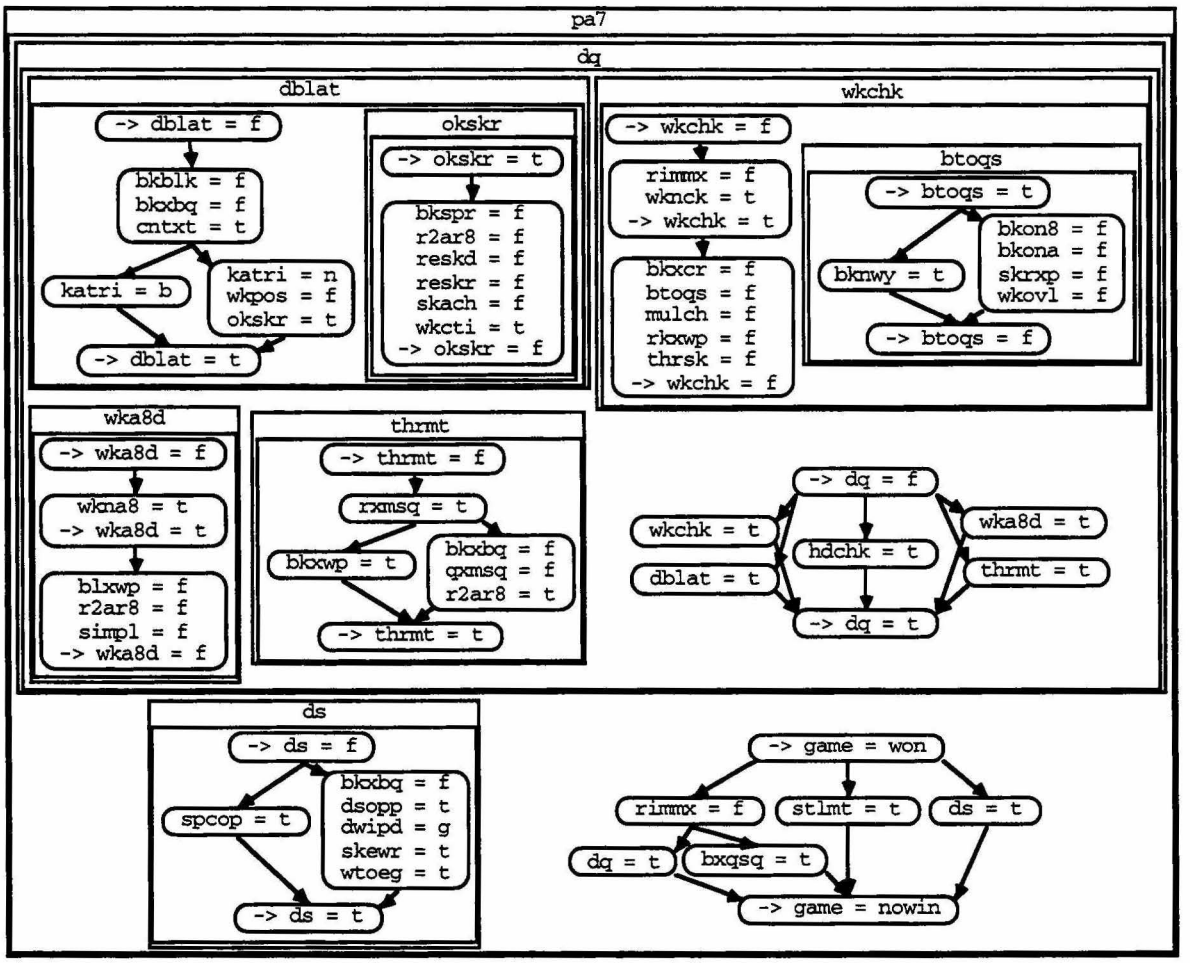


Figure 20 EDAGs induced for each of Shapiro's 9 sub-problems

Figure 21 shows the EDAGs of Figure 20 converted to a single EDAG by elimination of the intermediate decision variables. This is simple if the EDAG computing the variable has a single exception which is the truth value specified when the variable is used. There is one sub-problem, okskr, where the exception is single but the exception is the opposite to that required. The variable is eliminated by treating okskr = f as an exception in the EDAG for dblat. This involves adding two additional clauses (mulch = f, bkxcr = f) which correspond to alternative paths to nowin that the simple exception does not take into account. Similarly, the elimination of the wkna8 and wknck variables computed through multi-level exceptions requires the introduction of the additional variables marked by bullets.

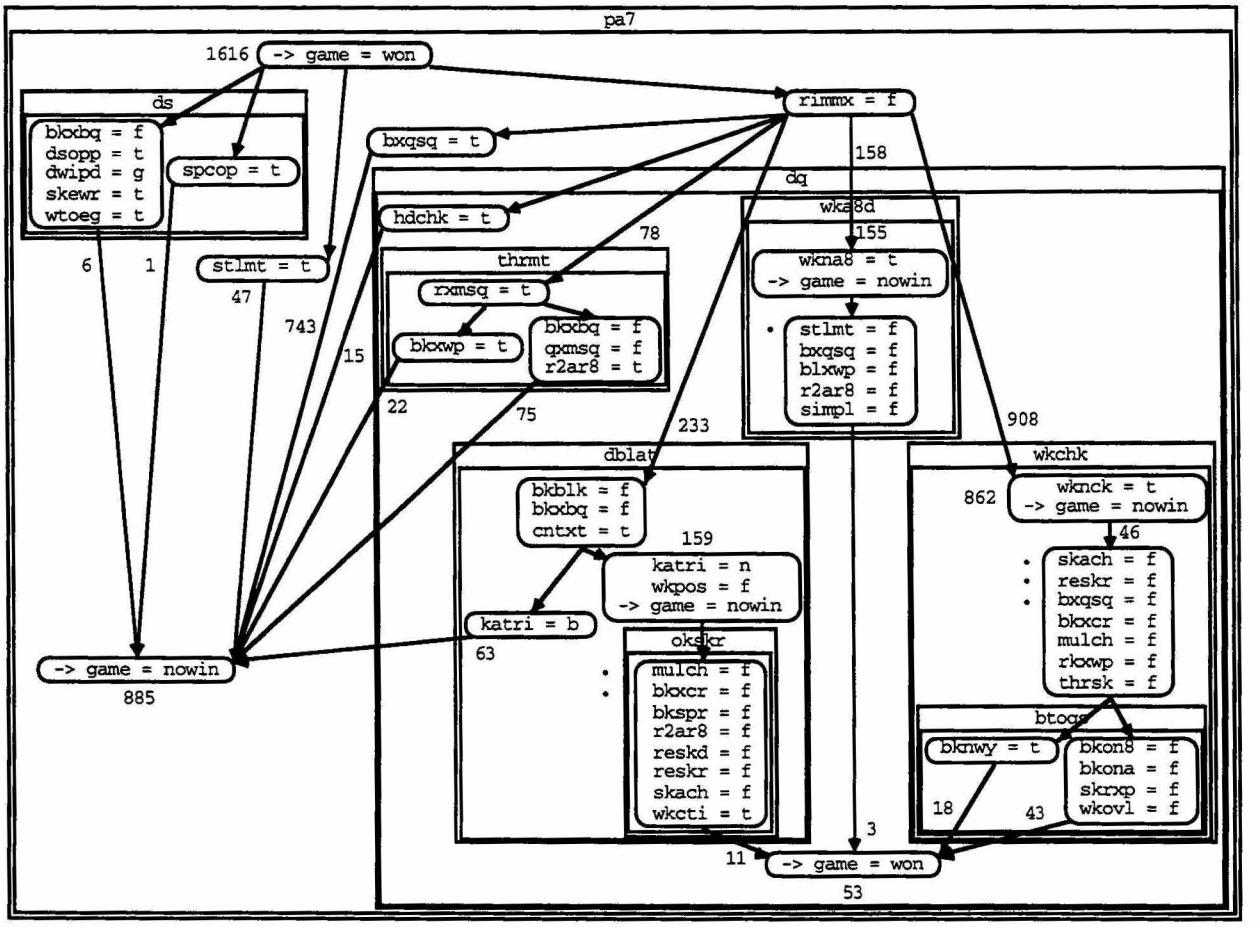


Figure 21 EDAG obtained from Figure 20 by eliminating the intermediate variables

Figure 21 is a single EDAG developed through structured induction that solves the KPa7KR problem. The numbers on the EDAG show the numbers of cases out of the total of 3196 that are covered by each part of the EDAG. It can be seen that the top level default accounts for 1616 cases, the path through scpop = t for 1 case, and so on. The numbers decided through all paths total substantially more than 3196 because a single case may flow through more than one path in an EDAG.

7.3 Comparing a Simply Induced Solution with that of Structured Induction

The EDAG of Figure 21 is a relatively simple exposition of the solution to what is said to be a difficult chess end- game situation, and it may be seen as a possible example of what Michie and Quinlan have been seeking, "a correct and effectively computable descriptions that can be

assimilated and used by a human being" (Quinlan, 1991). However, it was developed through a mixture of sub-problem structuring by human experts and inductive modeling. It is interesting to investigate whether a similar solution can be developed through inductive modeling without such human structuring.

Figure 22 shows the solution induced by Induct directly from the dataset of 3,196 cases. One difference from the EDAG of Figure 6 is that three of the attributes, bkxwp, rkxwp and stlmt, have not been used. Of these only stlmt seems to play a major role in Figure 6 accounting for 47 cases. However, if this path is removed only 2 errors occur so that 45 of the cases are covered by alternative paths. These 2 cases are covered by the rule wkna8 = t -> nowin in the EDAG of Figure 22.

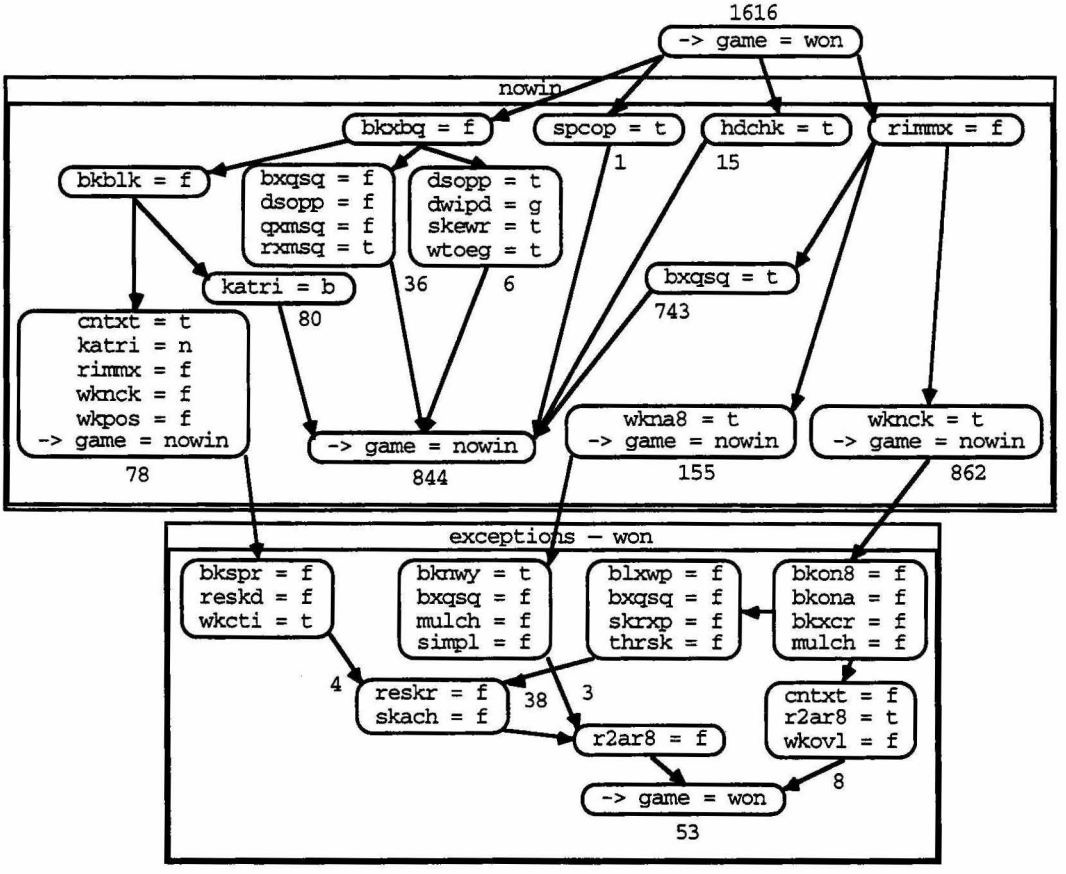


Figure 22 EDAG induced directly by Induct

Apart from the path through stlmt = t, it can be seen the top level structure above the exceptions in Figure 22 largely reproduces that of Figure 21. One difference is that some structures such as hdchk and dblat are under rimmx = f in Figure 21 and not in Figure 22. It turns out that it makes no difference to insert rimmx = f in these paths in Figure 22. It is a redundant clause that Induct eliminates.

Another difference is that the additional clause, wknck = f, has been introduced into the compound clause in ds. This corresponds to considering the delayed skewer threat only when the white king is not in check which is reasonable. It seems likely that the chess experts would have intended this when defining the skewer situation, but it was not taken into account when generating the cases for ID3.

The 53 exceptions to the top level structure are not generated through the same structures in Figures 21 and 22. The original okskr exception in dblat is reproduced exactly, but the exceptions relating to wkna8 and wknck are only loosely related in the two EDAGs. One possible reason for this is that there are two few cases for induction to determine the relevant attributes. In addition, these fairly rare and complex exception situations were simplified somewhat artificially in the Shapiro study using the ≤7 attributes and ≤1 screen of C criteria. They do not so closely correspond to basic chess concepts as do the upper level attributes. These exceptions and the differing knowledge structures generated need to be examined in detail by chess experts to determine precisely what is occurring.

In terms of the complexity measure defined in Section 5, the complexities of the EDAGs are: Figure 19, 70.6; Figure 20, 41.6; Figure 21, 30.4; Figure 22, 27.6. These figures and comparison of Figures 21 and 22 show that the unguided induction of Induct has generated a knowledge structure of similar simplicity to the guided induction of Shapiro's "structured induction".

8 EDAG REPRESENTATION AND INFERENCE TOOLS

EDAGs are simple to support through visualization and graphic editing tools interfaced to induction and evaluation systems. Figure 23 shows the Induct and KMap tools in KSSn (Gaines, 1994) being used to edit and evaluate EDAGs for the chess end game discussed above.

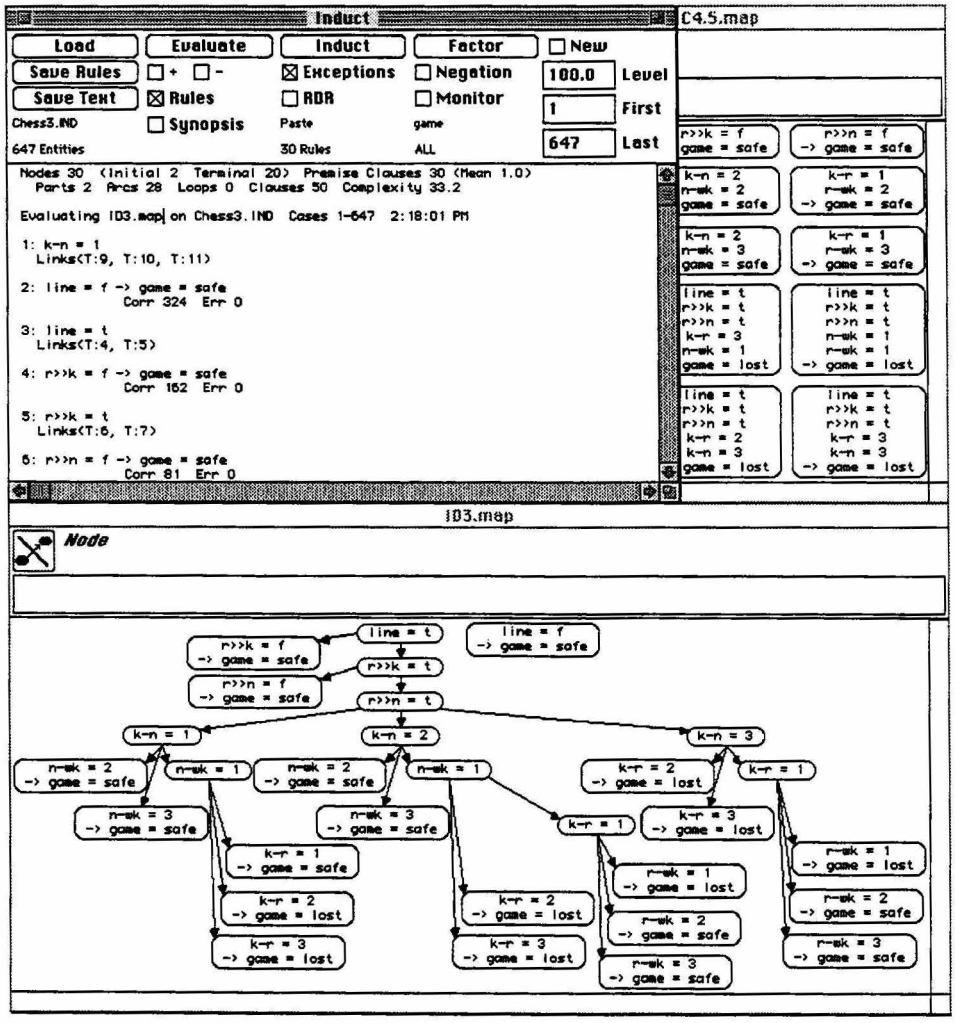


Figure 23 Graphic EDAG induction and editing tools

At the top left is an Induct server window with the chess data loaded. At the top right is a partial view of a KMap window with the C4.5 rules of Figure 11, and at the bottom is a KMap window with the ID3 rules of Figure 10.

KMap is a general visual language tool for knowledge structures (Gaines, 1991a) and concept maps (Gaines and Shaw, 1994). Induct exports its derived EDAGs as graphs in KMap, and edited graphs in KMap may be pasted back into Induct for evaluation of error rates.

The text in the bottom part of the Induct window at the top left of Figure 23 was generated by copying the EDAG in the KMap window at the bottom, pasting it into the Induct window, and then clicking the "Evaluate" button to cause Induct to evaluate the performance of the EDAG on the chess data.

The graphic interface between KMap and Induct makes it simple for an expert or knowledge engineer to experiment with the induced knowledge structures, investigate the roles of different parts, and attempt to improve the comprehensibility of what has been derived.

9 CONCLUSIONS

The factorization of systems of rules with exceptions is proposed as a means for simplifying induced rules to derive more comprehensible knowledge structures. The resultant structure, an exception direction acyclic graph, is a knowledge structure that subsumes both trees and rules. EDAGs make apparent the conceptual structures underlying inference through inheritance and structure sharing. EDAG inference is computationally simple and is also easily understood by people. EDAGs are simple to support through visualization and graphic editing tools interfaced to induction and evaluation systems.

The study of Section 8 shows that the knowledge structure produced by automatic induction of an EDAG can be very similar to that elicited directly from an expert. Clearly, many similar studies are necessary to substantiate these conclusions more generally, but the lack of datasets with expert associated expert models will make such studies slow to occur. Meanwhile EDAGs are of interest as potentially attractive alternatives to decision trees and rules.

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 grateful to Donald Michie whose unceasing drive to encourage the development of techniques for the induction of humanly comprehensible knowledge triggered much of the research on which this paper is based. Shapiro's end-game dataset was obtained from the Irvine machine learning archives managed by Patrick Murphy and David Aha.

REFERENCES

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. Van and Dufour, P., Ed. **Current Issues in Expert Systems**. pp.79-123. London, Academic Press.

Clancey, W.J. (1987b). **Knowledge-Based Tutoring: The GUIDON Program**. Cambridge, Massachusetts, 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. San Mateo, California, Morgan Kaufmann.

Gaines, B.R. (1991a). An interactive visual language for term subsumption visual languages. **IJCAI'91: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence**. pp.817-823. San Mateo, California, Morgan Kaufmann.

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

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

Gaines, B.R. (1994). Class library implementation of an open architecture knowledge support system. **International Journal Human-Computer Studies 41**(1-2) 59-107.

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.

Gaines, B.R. and Shaw, M.L.G. (1994). Concept maps indexing multimedia knowledge bases. **AAAI-94 Workshop: Indexing and Reuse in Multimedia Systems**. pp.36-45. Menlo Park, California, AAAI.

Holte, R.C., Acker, L.E. and Porter, B.W. (1989). Concept learning and the problem of small disjuncts. **IJCAI'89: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence**. pp.813-818. San Mateo, Morgan Kaufmann.

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

Muggleton, S. (1987). Structuring knowledge by asking questions. Bratko, I. and Lavrac, N., Ed. **Progress in Machine Learning**. pp.218-229. Wlmslow, UK, Sigma Press.

Murphy, P.M. and Aha, D.W. (1994). UCI Repository of machine learning databases. Department of Information and Computer Science, University of California at Irvine. ftp://ics.uci.edu.

Nosek, J.T. and 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.

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, Edinburgh 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.

Shapiro, A.D. (1987). **Structured Induction in Expert Systems**. Wokingham, UK, Addison-Wesley.

Shum, S.B. and Hammond, N. (1994). Argumentation-based design rationale: what use at what cost? **International Journal of Human-Computer Studies 40**(4) 603-652.