Some Results On the Computational Complexity of Refining Confidence Factors

Marco Valtorta
Department of Computer Science
University of South Carolina
Columbia, South Carolina 29208
(803)777-4641
mgv@cs.scarolina.edu (...!usceast!mgv)

Keywords: Knowledge base refinement, expert systems, knowledge acquisition, theory revision, concrete computational complexity, NP-Completeness.

Abstract

Rule bases are commonly used in the implementation of knowledge bases for expert systems. Knowledge engineers, with the help of experts, are good at designing the structure of the rule base, but poor at setting values for the strengths of the rules. This paper presents an algorithmic analysis of problems related to setting the strength of rules (synthesis) and to modifying them when good estimates are given (refinement). Models of knowledge base and learning protocols are discussed. We define several problems of rule strength synthesis and refinement and show them to be computationally intractable--even if we settle for approximate values or demand that cases be fitted only in a fixed percentage of cases. Perhaps the most striking result is that the refinement of rule strengths in MYCIN-like systems is intractable even when (1) the depth of rule chaining is bounded by two, (2) we are concerned only with the classification of outputs, rather than the exact values of the certainty factors of possible diagnosis, and (3) good estimates of the strengths are given. The concept of gradualness in knowledge refinement, introduced by Rada [31], is characterized using algorithmic complexity. Some problems in neural network learning are related to the ones introduced here. The paper ends by indicating directions for further research.

1. Introduction

Consider the following scenario: An expert system developer has implemented a rule-based expert system for student advising and has logged several dozen test cases. Each test case records a student's answer to the questions posed by the system, as the system attempts to provide sensible advice. Each case also records the system's advice and the advice that an experienced human adviser has determined to be the best for the student.

It would be convenient to have a program that modifies the rule base automatically, so that the expert system would give better advice, having learned from the human expert's performance on the collection of cases. It would be convenient, but how hard is it?

This paper reports on a formal investigation of how hard (in terms of computation time) it is for a rule-based expert system to learn, in the sense described in our scenario. The analysis is carried out using the techniques of concrete computational complexity. Overall, our results are discouraging. Most of the problems we define are shown to be NP-Complete or NP-Hard. This indicates that they cannot be solved in less than exponential time in the worst case.

The type of learning addressed in this paper is called *rule base refinement.* Knowledge base refinement involves the discovery of plausible refinements to a knowledge base that improve the accuracy of the associated expert system. Under this general definition, the knowledge base to be refined can be of any type. The definition does, however, limit the scope of knowledge base refinement to knowledge bases used by expert systems. It comes as no surprise, therefore, that most of the current work on knowledge base refinement is concerned with rule-based systems, the most common implementation of expert systems. This paper addresses knowledge refinement in the implementation phase of the knowledge-based system lifecycle, rather than in the analysis and design phases[1]. Concentrating on refinement in the implementation phase means that one is not limited to heuristic classification systems, even though that is the

[1]Conceptual models (Breuker et al. [4]) or domain-independent inference and tasks structures (Wilkins et al. [53]), which are produced during the analysis phase of knowledge-based systems construction, are likely to prove useful to knowledge base refinement, but related techniques are not considered in the body of this paper.

problem-solving task best addressed by rule-based implementations (Clancey [9]; Breuker et al. [4]).

A rule-based expert system is *truth-functional* (Ruspini [32]; Pearl [28]) if the belief associated with a proposition in the system depends only on the belief in propositions that appear in the premise of rules that conclude the original proposition[2]. This paper considers rule-based truth-functional systems. It seems advisable to gain an understanding of refinement in simpler knowledge-based systems, before moving to more comprehensive, but more difficult ones. One should note, for example, that computation of beliefs in intensional systems is, in general, known to be intractable (Cooper [10]), while computation of certainty factors in truth-functional systems is trivially easy.

There are four major dimensions in which a space of refinement procedures can be organized if only rule-based systems are considered. The first one is the model of the rule base itself. The second is the model of the rule interpreter. The third is the learning protocol, i.e., the source of information (typically, cases) that provides the input to the refinement procedure and the way in which information is presented to the refinement procedure. (In a more formal setting, Gold [20] calls this a "method of information presentation.") The fourth one is the feature (or features) of the knowledge base that is the object of the refinement procedure.

A point in this four-dimensional space will be described in the next section. The third section will discuss some refinement problems and prove that they are NP-Complete. Section 4 will consider an alternative learning protocol that stems from a different view of the role of a rule-based expert system, discuss related refinement problems and show that they are also NP-Complete. The fifth section will review the notion of gradualness in knowledge bases and attempt a formal characterization of it in terms of computational complexity. An efficient refinement algorithm is presented in this section. The paper concludes with a brief discussion of related work and of open problems.

---

[2]With an obvious exception for the propositions that are not concluded by any rule.

2. A model of a rule base and a learning protocol

In the model of a simple MYCIN-like rule interpreter and knowledge base used in this paper, a *rule* has the form IF (P1&P2&...&Pn) THEN C WITH STRENGTH A, where P1,P2,...,Pn and C are weighted propositions, and A, the *strength* of the rule, is a number in [0,1][3]. A *weighted proposition* is a statement, possibly true or false, with a certainty factor (CF). Certainty factors have values in [0,1]. A *combinator*[4] is a function from a vector of CFs that assigns a single CF to the conjunction of P1&P2&...&Pn of premises of a rule. A frequently used combinator function is MIN. The combinator output is multiplied by the strength of the rule to determine the CF associated with conclusion C of the rule[5]. Several rules may have the same conclusion. (Call it C.) Their collective input is merged by an *integrator* function that computes the CF associated with C in the rule base. Frequently used integrator functions are MAX and probabilistic sum[6].

It is convenient to represent rule bases as graphs with distinguished nodes corresponding to strengths, integrators, and attenuators. These graphs are called *inference nets.* For example, the rule base composed of IF P1 and P2 THEN P5 WITH STRENGTH A1 and IF P3 AND P4 THEN P5 WITH STRENGTH A2 would be represented as shown in Figure 1. Inference nets indicate the modifications of certainty factors due to combinators, strengths, and integrators,

Figure 1 An inference net representing two rules

---

[3]The model presented in this paper does not consider the MD (measure of disbelief) coefficients of MYCIN.

[4]A MYCIN analogue is $AND (Shortliffe [36, p.107]). Note that in this simple model, "predicate functions" such as MYCIN's THOUGHTNOT are not considered. See (Valtorta [40]) for relevant extensions to the model.

[5]This is done in MYCIN by CONCLUDE (or so it seems) (Shortliffe [36, p.132]; Buchanan and Shortliffe [7, p. 114]).

[6]The latter is used in MYCIN.

and allow a graphic definition of different structures of rule bases. In the example, the certainty factors of weighted propositions P1, P2, P3, and P4 enter input lines i1, i2, i3, and i4. These values are processed by combinators C1 and C2, strengths A1 and A2, and integrator I1. The output of integrator I1 is the certainty factor of weighted proposition P5.

Inference nets realize functions from vectors of CFs to vectors of CFs; a point in the graph of a function is called a *case*. In the example, the input vectors have cardinality 4, while the output vector consists of a single CF. A case for the net in the example is a pair composed of a vector of input CFs and an output CF.

The learning protocol used in this paper is the following. A set of cases is given. The cases are all available at the same time, rather than being provided in a specific (temporal) order. Of course, presentation of cases in a fixed order is a special case of presentation of cases as a set. Therefore, the complexity (e.g., run time) of an algorithm for refinement when a set of cases is given is a lower bound on the complexity of an incremental refinement algorithm.

The feature considered for refinement in this paper is the strength of rules. This decision reflects two conservative assumptions: that the structure of the rule base (i.e., the inference net) is fixed, and that the expert is more likely to be cooperative in tuning or debugging the rule base when it is fixed. This emphasizes the role of experts and knowledge engineers in *structuring* knowledge, which is consistent with the widely believed assumption that knowledge structuring is the key expert activity and is corroborated by the observation that rule strength refinement is a common activity in the practice of rule-based expert system construction, where it is sometimes called tuning (Gashnig [18]). Wilkins and Buchanan [46] argue that the structure of a rule base should be left unchanged even if the rules are obtained using machine learning techniques. An additional argument for selecting rule strengths as refinement features is that it is the easiest method for refining knowledge based systems[7].

[7]These arguments are not conclusive. The refinement of rule bases involving changes in their structure is likely to be sometimes necessary. Some relevant empirical work includes (Davis and Lenat [12]; Rada [30]; Rada [31]; Politakis [33]; Ginsberg [19]). The literature on refinement of expert systems written in Prolog and OPS5 is highly relevant when a numeric mechanism for uncertain reasoning is not used (Shapiro [33]; Brownston et al. [6]).

As argued in (Valtorta [40; 41]), it is unrealistic to expect experts to judge the correctness of single rule strengths when a rule is presented to them in isolation. This is independent of the semantics (or lack thereof) of rule strengths and is an inherent problem for rules that represent uncertain relations. There are no general techniques for estimating rule strengths precisely if modularity at the rule level must be maintained. For examples and additional arguments, see (Brooks and Heiser [5]; Valtorta [39;40]; Wilkins and Buchanan [46]; Heckerman and Horwitz [24], Pearl [28, pp.8-10]). For an impractical rule refinement technique that does not adhere to the claim, see (Shapiro [34]).

3. Some NP-Complete refinement problems

Given the choices of knowledge base model, learning protocol, and refinement feature given in section 2, some additional choices still need to be made to define a problem in a semi-formal notation, such as the one used by (Garey and Johnson [17]. Specifically, the topology of the inference network (chain, tree, or acyclic graph) and the combinator and integrator functions still have to be chosen. In this section, the network is a tree. This corresponds to a rule base where the premises of all pairs of rules do not share propositions. Of course, the complexity of rule base refinement in trees is a lower bound of the complexity in graphs. In the first result proven in this section, the combinator and integrator functions are MIN and MAX respectively, uniformly throughout the tree. In the second result, these functions are MIN and Probabilistic sum. A third result is given without proof.

For expository reasons, the proven results concern a synthesis, rather than a refinement problem. In a *synthesis* problem, no estimates of rule strengths are given by the expert[8]. For the proof that follows, trees are restricted in such a way that the in-degree of combinators is 2, while the in-degree of integrators can grow with the size of the network. The restriction on in-degree of combinators means that each rule in the rule base can have only up to two weighted propositions in its premise. On the other hand, the absence of restrictions on the in-degree of

[8]Synthesis of rule strengths is a kind of refinement of rule bases.

integrators means that there can be any number of rules concluding the same weighted proposition. Also, the height of the inference tree is bounded by a constant (fixed number, independent of the size of the net). Combinators and integrators must be alternating, i.e, there is no edge in the inference net whose endpoints are a pair of combinators or a pair of integrators.

These restrictions model certain observed properties of classification rule-based expert systems: they typically exhibit short inference chains, while their growth consists mostly of the addition of rules that conclude intermediate, already present, propositions. Whether one agrees with the typicality of this observation or not, the complexity of strength synthesis in restricted trees is a lower bound on that for unrestricted trees.

Finally, the problems that we consider in this section restrict the possible values of rule strengths to be 0 or 1[9].

For the purpose of the proof that follows, it is sufficient to consider nets with the topology described in Figure 2.

We note that this topology does not have MINs as leaf nodes. This is somewhat unsatisfactory, because one would expect to see the net composed of building blocks like that in Figure 3.

Figure 2 Restricted nets

[9]Therefore, these problems are not special cases of problems for which rule strengths are allowed to vary in the interval [0,1], in the sense that the instances of the latter are not necessarily also instances of the former. However, by extending the proofs we give in this section, it has been shown that the latter problems are just as hard (Valtorta [40]). The extension is somewhat technical for the first proof, while it is trivial for the second proof, as will be noted in due course.

Figure 3 A natural building block

Each of the blocks in Figure 3 corresponds to rules concluding the same proposition. Still, the net in Figure 2 can be viewed as a "shorthand" for the more "natural" inference net in Figure 4. More precisely, it will be shown that the input to the network in Figure 4 can be selected to simulate any desired assignment of inputs to the network in Figure 2. Consider an n- input network of the type in Figure 2 and a 2n-input network of the type in Figure 4. Consider an arbitrary assignment of inputs (1 through n) to the network in Figure 2. Let the even inputs to the network in Figure 4 have input value 1 and the odd input j have the value of the input $(j+1)/2$ to the network in Figure 2. Clearly, because of simple properties of the MIN function, this mapping guarantees that the output of the network in Figure 4 is the same as the output of the network in Figure 2.

After all these preliminaries, a synthesis problem can be stated as follows.

Figure 4 Net equivalent to that of Figure 2. (See text for details.)

Problem name: Restricted Rule Strength Synthesis (RS).

Problem instance: An inference tree with alternating MIN and MAX nodes, multiplicative rule strengths with value 0 or 1 at the output of MIN nodes, maximum fan-in to MIN nodes equal to 2, maximum height equal to 3; a set of cases.

Question: Is there an assignment of strengths for which the function realized by the inference net satisfies the cases?

Theorem 1. RS is NP-Complete.

Remark: The corresponding search problem is therefore NP-Hard. ("Find an assignment of strengths for which the function realized by the inference net satisfies the cases.")

Proof

Membership in NP is trivial. The non-deterministic program that solves the problem has a loop whose body is an assignment of 0 or 1 (non- deterministically) to each rule strength.

We transform Monotone 3-Conjunctive Normal Form Satisfiability (MSAT) (Garey and Johnson [17]) to RS. Recall that the generic MSAT instance is a propositional logic formula in conjunctive normal form, where each clause contains only three negated *(negative clause)* or three unnegated *(negative clause)* variables (e.g., (x1vx2vx3) & (-x1v-x4v-x5)). The question is whether there is a satisfying truth assignment for the expression.

Given an expression E of MSAT, the following algorithm produces, in time polynomial in the size of E, an instance of RS such that the Question has answer yes if and only if E is satisfiable.

Let n be the number of distinct variables in E, m be the number of clauses in E. (n and m can be obtained in polynomial time from any reasonable encoding of E.)

The tree of (the) RS (instance) is shown in Figure 5, where we also show the numbering scheme for the input rule strengths.

There are 2n+ m cases for the RS instance: Here is how to build them.

(a) m *clause cases.* Each one of these cases corresponds to a clause. Each of these cases has

Figure 5 Tree of the generic RS instance

output .6. Each of the inputs that is not explicitly specified in the following has value 0. If the clause is positive and contains the variable vi, vj, vk, the case is built with: .6s for inputs i, j, k, .9 for input 4n+ 2. If the clause is negative and contains the variables vi, vj, vk, the case is built with .6s for inputs 2n+ 1+ i, 2n+ 1+ j, 2n+ 1+ k, and .9 for input 2n+ 1. Intuitively, one can say that the left MAX box is for positive clauses, while the right MAX box is for negative clauses.

(b) 2n *variable cases*. Each pair of these corresponds to a variable. The first case of each pair has output 0. The second has output .6. Each of the inputs that is not explicitly specified has value 0. For variable vi, the first case of each pair has value .6 for inputs i and 2n+ i+ 1; the second case has value .6 there and .9 in positions n+ i and 2n+ 1+ n+ i.

These cases can clearly be built in time polynomial to n, the number of variables in E[10].

An example of how MSAT instances are mapped into RS instances is given in Figure 6. With reference to the figure, T1 and T2 are the clause cases, T3 and T4 are the variable cases corresponding to x1, T5 and T6 are the variable cases corresponding to x2, and T7 and T8 are the variable cases corresponding to x3.

It will now be shown that the instance of RS built according to the above algorithm is (A) a yes-instance if E is satisfiable and (B) a no-instance if E is not satisfiable.

Lemma 1. Consider the variable cases corresponding to a generic variable vi. The first case in each pair is satisfiable in isolation if and only if at least one of the strengths in positions

[10]Everywhere in this proof, any pair of values (a,b) can be used in place of (.6,.9), as long as 0< a< b< 1. Moreover, each clause case and each pair of variable cases can use a different pair of values.

Figure 6 Instance of RS corresponding to $(x_1 v x_2 v x_3) \& (-x_1 v -x_2 v -x_3)$

i and 2n+ 1+ i is 0, because the output of the MIN box is 0 if and only if at least one of its inputs is 0. The second case is satisfied in isolation only if at least one of the strengths in positions i and 2n+ 1+ i is 1, because the output of MIN is .6 only if at least one of its inputs is .6. Therefore *exactly one* of the two strengths is 1 and the other is 0. (Note that the variable cases do not specify *which* of the strengths is 1 and which is 0.)

Lemma 5.2 Consider the clause cases. Consider the case corresponding to a generic clause, consisting of three literals, li, lj, lk. Assume that the literals are positive. The case is satisfied in isolation only if one of the strengths in position i, j, k is set to 1, because the output of the MIN box is .6 only if at least one of the inputs is .6. The case for negative clauses is similar, but the strengths are now in positions 2n+ 1+ i, 2n+ 1+ j, 2n+ 1+ k.

(A) If E is satisfiable, then RS is a yes-instance. In order for RS to be a yes-instance, there must be some assignment of strengths such that all cases are "satisfied," i.e., the case input must produce the case output. Assume that each clause in E is satisfiable, i.e. the case

inputs must produce the case output. Consider a generic clause first. Set to (0,1) the two strengths in position i and 2n+ 1+ i, respectively, if variable vi is T in E's model; set the strengths to (1,0) if variable vi is F in E's model. Since E has a model, it cannot be that such a pair of strengths is mapped into (0,1) and (1,0) simultaneously. Therefore, by Lemma 1, the variable cases are satisfied. Now, note that each clause must be satisfied in order for E (in CNF) to be satisfied. Therefore, a positive clause[11] will have a variable set to T in the model for E. This means that the corresponding case has .6 as output, because one of the .6s from the left MAX box will have the corresponding strength set to 1, and, being MINned with the .9 from the right MAX box, it will propagate to the output. Therefore the case is satisfied.

(B) If RS is a yes-instance, then E is satisfiable, i.e., E has a model. Given RS, build the model according to the following correspondence: if the strength in position i has value 1 and the strength in position 2n+ 1+ i is 0, then variable vi has value T; if these strengths have values 0 and 1, then variable vi has value F. As shown in Lemma 1, (0,1) and (1,0) are the only two possible cases. Since E is in conjunctive normal form, E is satisfiable if (and only if) each clause is. Each clause is satisfiable if each clause case is, as shown in Lemma 2.

(End of proof.)

It will now be shown that rule strength synthesis is NP-Hard for inference trees where MIN is used as the combinator function and probabilistic sum is used as the integrator function. This result implies that rule strength synthesis in a MYCIN-like system is NP-Hard.

Problem name: Restricted Rule Strength Synthesis, MIN/p+ case (RSP).

Problem instance: An inference tree as described in Figure 7[12], where rule strengths have value 0 or 1; a set of cases.

Question: Is there an assignment of strengths for which the function realized by the inference net satisfies the cases?

---

[11]The argument for negative clauses is analogous.

[12]The strength feeding directly into the MIN node can be considered a shorthand for a degenerate, 1-input p+ node with attenuated output.

Figure 7 Tree of the generic RSP instance

Theorem 2. RSP is NP-Complete.

Proof

Membership in NP is trivial. The non-deterministic program that solves the problem consists of a loop whose body is an assignment of 0 or 1 (non deterministically) to each strength.

One in Three Satisfiability (OTS) (Garey and Johnson [17, p.259]) will be transformed into RSP. The variant in which no clause in the formula contains a negative literal will be used. The generic OTS instance is a formula in conjunctive normal form, with no negative literal. The question is whether there is a model for the expression such that each clause has exactly one true literal.

Given a formula E in monotone 3-conjunctive normal form, the following algorithm produces in time polynomial in the size of E an instance of RSP such that the Question has answer yes if and only if E has a model in which only one literal per clause is true.

Let n be the number of distinct literals in E, m be the number of clauses in E. (n and m can be obtained in polynomial time from any reasonable encoding of E.)

The integrator (p+) node in the instance of RSP has n inputs. There are 2m cases: two cases correspond to each clause in E. Let a generic clause contain the variables vi, vj, vk. The first case for each clause has value .6 corresponding to inputs i, j, k, and value .7 corresponding to the rightmost input to the net in Figure 6. When .8 is substituted for .7 and .7 is substituted for .6, the second case for this clause is identical to the first one.[13]

[13]The triple (.6,.7.,.8) can be replaced with (a,b,c), as long as 0< a< b< c< 1.

As an example, Figure 8 shows the instance of RSP corresponding to E = (x1vx2vx3) & (x2vx3vx4). In the Figure, T1 and T2 correspond to the first clause in E, while T3 and T4 correspond to the second clause in E.

An instance of RSP built according to the algorithm just given is a yes-instance if and only if the corresponding instance of OTS is a yes-instance.

The "if part" is simpler. If a variable in the model for E is T, set the corresponding strength to 1. Otherwise (except for the rightmost strength, which is always 1), set strengths to 0. This insures that exactly one of the strengths input to p+ and corresponding to non-0 input case values is 1 for each case. Therefore, the non-0 (.6 or .7) input case value propagates unchanged to the output. Therefore, each case is satisfied.

The "only if" part is proved now. It will be shown that if RSP is a yes- instance, then necessarily OTS is a yes-instance. Assume that RSP is a yes-instance. It will be shown that in order for RSP to be a yes-instance, it must be that exactly one of the strengths corresponding to the p+ node inputs is 1 for each case. By assigning T to the variable corresponding to this unique strength, a model for E is obtained which satisfies the "one in three" condition. Con-

Figure 8 Instance of RSP corresponding to $(x_1 v x_2 v x_3) \& (x_1 v x_2 v x_4)$

sider a generic pair of cases corresponding to a clause in E. We show, by algebraic manipulation, that this pair is satisfied if and only if exactly one of the three strengths corresponding to the cases is 1. Call the strengths x, y, and z and let [p+] indicate the probabilistic sum operation. The pair of cases is satisfied if and only if the following system has a solution:

.6x [p+] .6y [p+] .6z = .6

.7x [p+] .7y [p+] .7z = .7,

i.e., after carrying out the probabilistic sums,

x + y - .6xy + z - .6xz - .6yz + .36xyz = 1

x + y - .7xy + z - .7xz - .7yz + .49xyz = 1.

If any two of the variables have value 0, the system has a solution if and only if the other variable has value 1.

To show that the system has no solution if only one of the three variables is 0, subtract the second from the first equation, side by side:

xy + xz + yz = 1.3xyz

This equation has no solution if only one of the three variables is 0.

The only case left is that in which the three variables are all positive (and, of course, no greater than 1). In this case, each of the products xy, xz, and yz is greater than or equal to xyz:

xy + xz + yz > 2xyz > 1.3 xyz, and therefore it is impossible that

xy + xz + yz = 1.3xyz.[14]

(End of proof.)

Some truth-functional systems (e.g., MIDST by Biswas and Anand [2]) use Dempster's rule as an integrator. Dempster's rule is described by, e.g., Pearl [28, chapter 9] and Gordon and Shortliffe [21]. It is more often used in intensional systems. It will now be shown that

[14]The second part of the proof, unchanged, shows that strength synthesis is NP-Hard when rule strengths are allowed to vary between 0 and 1 (inclusive), rather than being restricted to 0 and 1.

rule strength synthesis is NP-Hard for inference trees where MIN is used as the combinator function and Dempster's rule is used as the integrator function[15].

Problem name: Restricted Rule Strength Synthesis, Dempster's rule case (RSP).

Problem instance: An inference tree as described in Figure 7[16], where rule strengths have value 0 or 1; a set of cases.

Question: Is there an assignment of strengths for which the function realized by the inference net satisfies the cases?

Theorem 3. RSD is NP-Complete.

Proof

As shown in section 3.3 of (Gordon and Shortliffe [21]), Dempster's rule provides the same result as would the probabilistic sum function, since all rules corresponding to strengths s1,...,si conclude the same proposition. Therefore, the proof that RSP is NP-Complete is also a proof that RSD is NP-Complete.

(End of proof.)

Synthesizing *approximate* values for rule strengths is also NP-Hard, as shown by the following result. (See (Valtorta [40]) for the proof of the result.)

Problem name: Approximate Restricted Rule Strength Synthesis (ARS).

Problem instance: An inference tree as described in Figure 2, with strengths in [0,1]; a set of cases.

Question: Is there an assignment of strengths for which each strength is within less than 0.5 from the correct strength?

ARS is NP-hard.

---

[15]Suggested exercise for the reader: show that the result holds when product is used in place of MIN.

[16]Take p+ to stand for an application of Dempster's rule, usually indicated using the exclusive or ("plus in a circle") symbol in the literature.

A rule strength refinement problem and a related result are now stated, after a few necessary definitions. See (Valtorta [40]) for the proof of the result and for additional considerations.

Given an inference net, the *error* on a case is the difference between the output part of the case and the output CF (CFs) obtained by propagating the input part of the case through the net (graph).

Problem name: Approximate epsilon refinement (AER).

Problem instance: An inference tree such as the one in Figure 2; a set of cases; a constant e; an assignment of (expert-given) strengths, each of which is at most e away from the correct one; a constant q.

Question: Find an assignment of strengths

(a) within e of the expert-given ones

(b) for which the error on each case is no greater than using the expert- given strengths

(c) for which the error on at least one case is q less than using the expert-given strengths, or a non-zero error is reduced to zero.

This search problem is NP-Hard.

4. Rule-based systems as classifiers

In this section we make assumptions about the *task* (Clancey [9]; Breuker et al. [4]) performed by the expert system. Consider that the rule-based expert system is used as a classifier.

In some applications, users expect to obtain from the expert system some "confidence" associated with a classification. For example, consider the following scenario. The output of a diagnostic system is used as the input to a treatment system, which is required to cover the most likely diseases and must take into account additional factors, such as side effects or costs of treatments. Some indication of the relative likelihood of each disease would clearly be

[17]It is (very conservatively) assumed that the existence of a correct assignment of strengths is known, i.e. that the inference net handles the cases correctly for a particular assignment.

useful in determining an appropriate treatment[18]. To state this differently, the diagnosis (classification) can be viewed as having the role of a solution, in the diagnostic system in isolation, or of a hypothesis, in the combined diagnosis-and-treatment system. Since the same mechanism of certainty factors is used in both cases, one must conclude that training cases for the diagnostic system should include a certainty factor. This is consistent with the use of certainty factors in MYCIN as described by Adams: "The certainty factor is used in two ways. One is to rank hypotheses to select those for further action. The other is as weighting factor for the credibility of a hypotheses (...)"[19] (Buchanan and Shortliffe [7, p. 269]).

In other applications, it may not be realistic to assume that cases (as defined in section 2) are given precisely. Instead, in classification problem solving, it is more likely that the correct class be known for a given description, without any indication of the certainty factor associated with the class. This observation is based on the fact that the task of the expert system is to classify cases. The certainty factor mechanism is viewed only as a method to achieve a correct classification: the absolute values of certainty factors of a class are irrelevant; all that counts is that the class with the highest certainty factor be selected as the best one. Two problems related to classification will be defined and analyzed in the remainder of this section.

Consider first the situation in which a set of cases is given, but all that is required of the inference tree is that the *order* between output values of each pair of cases, rather than the exact value of the cases be maintained by the inference network with synthesized weights. Here is the formal problem statement.

Problem name: Restricted Rule Strength Synthesis, ordered output case (RSO).

Problem instance: The same as for problem RS (in the previous section).

---

[18]The view of diagnostic expert systems as simple classifiers is held in the work on knowledge acquisition and refinement at Rutgers University (Drastal and Kulikowski [13]; Politakis [29]; Ginsberg [19]), while the view that diagnostic expert systems can be trained to output a usable certainty factor as well as a class and that cases include certainty factors is held in (Rada [30,31]; Fu and Buchanan [15]; Caruana [8]; Valtorta [40-42]).

[19]Adams continues to show that a diagnosis with a higher certainty factor is not necessarily more probable than one with a lower certainty factor, and therefore that MYCIN behaves irrationally, in the sense of (Buchanan and Shortliffe [7, chapter 11]).

Question: Is there an assignment of strengths for which the order of the output parts of each case is the same as the order of the outputs of the inference net when input the input part of the same case?

Theorem 4. RSO is NP-Complete. *Proof.*

The proof that RS is NP-Complete is, unchanged, also a proof that RSO is NP-Complete.

(End of proof.)

Now consider the case in which the inference net has multiple output lines, but the subnets rooted in each of the output lines are trees. (See Figure 9 for an example.) In the following problem, the inference net is a simple classifier, as discussed in Section 3. Each case consists of input vector (of CF's) and a class. The class computed by the inference net is the identifier of the output line with the highest CF[20]. For the purpose of the proof that follows, it is sufficient to consider inference nets with two outputs, of the form given in Figure 10[21].

Figure 9 Inference net with multiple outputs

Figure 10 Net of the generic RSM instance

[20]Ties must be handled in some fashion. (See (Politakis [29]) for relevant considerations.) The proof below does not depend on the way ties are handled.

[21]As was the case with Figure 2, this is a somewhat "unnatural" (but very simple) inference network. Again, a slightly more complex and more natural inference net can be mapped into this one.

Problem name: Restricted attenuation synthesis, multiple outputs (RSM).

Problem instance: An inference net as described in Figure 10, where rule strengths have value 0 or 1; a set of cases as defined in this section.

Question: Is there an assignment of strengths for which the function realized by the inference net satisfies the cases?

Theorem 5. RSM is NP-Complete.

Proof

Membership in NP is trivial. The non-deterministic program that solves the problem has a loop whose body is an assignment of 0 or 1 (non-deterministically) to each rule strength.

We transform Restricted Rule Strength Synthesis (RS, shown to be NP-Complete in section 3) to RSM. The reader is requested to infer the algorithm that maps instances of RS into instances of RSM from the simple example illustrated in Figure 11. Figure 11 shows the instance of RSM corresponding to the instance of RS in Figure 2.

(End of proof.)

5. Gradualness

Rada argued that gradualness facilitates knowledge refinement. He defined *gradualness* as meaning that "small changes in the structure of the knowledge correspond to small and meaningful changes in its function" (Rada [31]). In this section the meaning of gradualness is reconsidered in the light of the NP-Hardness results for rule strength synthesis and refinement. Some needed definitions will be introduced first. Except where noted, only the MIN/MAX case will be considered.

A *choice box* is a function whose output is equal to one of the inputs.

MIN and MAX are examples of choice boxes.

Figure 11 Instance of RSM corresponding to the instance of RS in Figure 6

A *winner* at a choice box is the input value that is equal to the output value. (In the case of output values that are equal to more than one input value, the winner is chosen arbitrarily among these inputs.)

A *setting of winners* for an inference net is an assignment of winners for all choice boxes in the inference net.

From now on, only inference nets where integrators and combinators are 2-input choice boxes will be considered[22]. A *loser* at a 2-input box is the input value that is not equal to the output value. In the case of a tie, the loser is chosen to be the other value with respect to the arbitrarily chosen winner.

A *switch setting* for an inference net with 2-input boxes is a setting of winners for the inference net.

---

[22]Valtorta [40] shows that the results extend to n-input boxes, for any n > 1.

The NP-Hardness proof for AER does not carry through if the problem instance is augmented to include as input a switch setting for the tree. This observation leads to the definition of the following search problem.

Problem name: Rule strength synthesis, no-switch case (RSN).

Problem instance: An inference tree like the one in Figure 2, strengths in [0,1]; a set of cases CS= {C1,...,Cj}; a switch setting for each case in CS.

Question: Find an assignment of strengths compatible with the switch setting and such that the cases are satisfied, if such an assignment exists.

The algorithm that will be described now solves RSN in $O(n^2 j \log(nj))$, where n is the number of strengths at the inputs[23] to the inference tree. (Name the n strengths s_i, $0 < i < n-1$).

The algorithm has an instance of RSN as input and returns an assignment of strengths, as requested in the question for RSN. It ends in failure at step 2, if such an assignment does not exist.

1. For each case, Ti, and each choice box, set up a 2-variable linear inequality as follows: let $s_{i1} * i_{i1}$ be the winner at the choice box; let $s_{i2} * i_{i2}$ be the loser. If the box is a MAX (MIN) box, let the inequality be $s_{i1} * i_{i1} >= (<=) s_{i2} * i_{i2}$. Add 2n inequalities of the form $s_i >= 0$, $s_i <= 1$, i= 1,...,n. (There are at most 3n-1 inequalities for each case, since there are n-1 choice boxes in a complete tree with n inputs.)

2. Solve the system of linear inequalities obtained in step 1[24].

This algorithm indicates a way to exploit expert-given strengths in refining rule strengths. The expert-given strengths to be refined should be used to compute winners at each box for each case. The inequalities are set up assuming that winners do not change when strengths are changed from the expert-given strengths to the correct ones. If the expert-given strengths are accurate predictors of winners, this technique is successful.

---

[23]All strengths internal to the tree can be set to one (i.e., eliminated) without loss of generality, as shown by Valtorta [40-42].

[24]To guarantee the run-time bound given above, step 2 should be solved using the algorithm by Megiddo [27].

An example of use of this algorithm on the inference net of Figure 12 is the following. Let the expert-given strengths be (.5,.1,.4,.4). The case set consists of the single case T1= ((.6,.5,.8,.2),.1). Assuming that winners at each choice box do not change with respect to the ones obtained by running the input part of the T1 through the network, one has the following system of 7 inequalities and one equality:

$.8x_3 - .1 = 0$

$.6x_1 <= .5x_2$

$.8x_3 <= .2x_4$

$.8x_3 >= .5x_2$

$0 <= x_i <= 1,\ i = 1,...,4.$

This system has an infinite number of solutions, e.g., (.5,1,1/8,.4).

It is an open question whether rule-based expert systems are designed in such a way that strengths are reliable predictors of winners[25]. Empirical results (Politakis [29]; Rada [31]; Caruana [8]; Ginsberg [19] indicate that they are.[26]

Unfortunately, reliable prediction of winners cannot be the exclusive reason for successful refinement, because the problem corresponding to RSN when probabilistic sum is used in place of MAX is to be NP-Hard, as shown below.

Figure 12 Inference tree for example in section 6

[25]Because of some underlying principle of knowledge organization?

[26]There may be some other as yet unknown reason why rule-base refinement is reported to work effectively in controlled experiments with pre-existing rule bases.

Problem Name: Rule strengths synthesis with probabilistic sum, no-switch case (RSNP).

Problem instance: An inference tree like the one in Figure 7, strengths in {0,1}; a set of cases; a switch setting for each case.

Question: Is there an assignment of strengths compatible with the switch setting and such that the function realized by the inference net satisfies the cases?

Theorem 6. RSNP is NP-Complete.

Proof (Sketch)

All the instances of RSP used in the proof that RSP is NP-Complete are also instances of RSNP when the switch setting given for each case indicates that the left branch entering the MIN box is the winner.

This result indicates that there is a fundamental computational difference between the use of MAX and the use of probabilistic sum as integrators.

6. Conclusions

6.1. Related Work

6.1.1. Some related problems in neural network training

Inference nets resemble feed-forward neural nets. Judd [25[27]] states a learning problem for neural networks as follows: "Given a neural network and a set of training examples, does there exist a set of edge weights so that the network produces the correct output for all the training examples?"

Judd's problem is analogous to a rule strength synthesis problem. The model used in his work clarifies this relationship. Figure 13 is an example of feed-forward net presented by Judd [26]. The learning problem is to find functions for a, b, and c, such that the output (f) of the

[27]Historical note: the result that RS is NP-Complete dates to January 1985. The author did not know of Judd's result until October 1987.

Figure 13 A neural network (adapted from Judd [26])

net handles given cases. (Judd calls the cases "data.") This is a more general setting than the one used elsewhere in rule strength synthesis and refinement, since the functions for a, b, and c are not fixed *a priori*, while we fix integrators and combinators. However, Judd's NP-completeness result is specifically obtained for a net where the functions are restricted to OR and AND over bits. Restrictions on the topology of the network are different from the ones considered in this paper. A full analysis of the differences is beyond the scope of this paper.

Blum and Rivest [3] use a similar approach, and obtain NP-Completeness results for different network topologies and different node functions.

In general, training a neural network is analogous to synthesizing strengths in a rule-based expert system, and it should be expected that algorithms and complexity results be similar. In neural networks there is no process analogous to knowledge acquisition from experts. Therefore, rule strength refinement seems to be peculiar to expert systems[28].

Conversely, Fu [14] has proposed using the back-propagation algorithm to synthesize strengths in expert systems for certain choices of integrator and combinator functions.

6.1.2. Related models

The assumption that the expert system is a black box is common to all work in knowledge base refinement. Dispensing with this assumption greatly simplifies rule strength synthesis (Valtorta [40, chapter 8]) but does not seem suitable for realistic applications since, as Fu [14] observes, "expert knowledge for the most part concerns the associations between observable data and final hypotheses."

---

[28]The presence of the expert constrains rule-base refinement. For example, it is desirable that refined strengths be "as close as possible" to expert-given ones.

The use of linear functions as integrators and combinators greatly simplifies rule strength synthesis (Gallant [16]) but seems hardly justifiable in terms of any theory of belief. The existence of intermediate possibilities is an open problem.

Case-based, memory intensive expert system implementations may be easier to debug and refine. See (Sharma and Sleeman [35]) for a discussion.

Valtorta [42] explores the use of oracles as an alternative learning protocol. An oracle outputs the output part of a case when given the input part. It corresponds, intuitively, to the situation in which a perfect expert is on call to answer questions concerning the input/output behavior of the rule-based system. Some NP-Hard refinement problems become tractable when an oracle is available.

6.2. Summary of results

This paper outlined some complexity results on knowledge base refinement. The results are of a theoretical nature. The reader should appreciate that an "algorithms and complexity" analysis of the kind presented is a useful complement to more empirical or statistical analyses as described by Politakis [29], Ginsberg [19], Craw and Sleeman [11] and others [43]. The main results presented in this paper are proofs of NP-Completeness and NP-Hardness for various kinds of refinement problems.

Most of the results presented are independent of the task performed by the expert system. However, they hold when the expert system is viewed as a classifier (cf. section 4).

The complexity results indicate that it is unlikely that implementation level refinement will lead to the user's dream of expert systems that "refine or add to the knowledge base as the intelligent system has new experiences" (Hafner [22]).

Our results hold for on-line (incremental) as well as off-line algorithms. In fact, any algorithm that tries to refine the rule base as each case is available is doomed to be no more efficient that an off-line algorithm.

Our results hold for very shallow rule bases with a very small number of conditions in each rule premise. Additional restrictions on the topology of inference networks seem meaningless, and it is safe to conclude that the complexity of rule base refinement does not derive from the depth of rule chaining.

Our results hold when the possible values of rule strengths are restricted to be in a very small set. Additional restrictions on the alphabet of rule strenghts seem meaningless, and it seems safe to conclude that the complexity of rule base refinement does not derive from the size of the alphabet of possible rule strengths.

6.3. Directions for future work

The analogue of strengths in rules are probability masses (Smets [37]) in Dempster-Shafer belief networks and conditional probabilities (or related parameters, such as likelihood ratios) in Bayesian networks (Pearl [28]). These numerical parameters, just like rule strengths, can be estimated using statistical techniques. However, just like rule strengths, they are often estimated using knowledge engineering techniques. Consider MUNIN, one of the largest expert system developed using the Bayesian network approach. As of mid-June 1989, MUNIN had about 1000 nodes[29]. "Estimating the 270 conditional probabilities (...) would require at least 10000 cases. Instead of relying on this empirical approach, we have tried to rely as much as possible on 'deep knowledge,' using the understanding of patophysiological processes as expressed in medical textbooks and papers" (Andreassen et al [1, p.369]). In any case, the numerical parameters need to be refined. In the case of MUNIN, they are said to be "revised": "Discrepancies between the network and the medical experts lead to revision of the model parameters and occasionally to revision of inaccurate or incomplete models" (Andreassen et al [1, p.370]). See (Valtorta [44]) for initial results on the refinement of probability masses in a kind of intensional system.

Some methodologies and tools for knowledge engineering emphasize the use of models of expertise at different levels. Most notable among them is KADS (Hayward et al. [23]). Since

[29]Steen Andreassen, personal communication. MUNIN has been developed in the context of ESPRIT project 599.

refinement at the implementation level is, in general, intractable, it is warranted to explore the possibility (and efficiency!) of refinement at the analysis and design levels or, equivalently, during earlier stages of the KBS lifecycle. Meta-knowledge and records of project history may prove useful in refining an implemented expert system. This was partly shown in TEIRESIAS (Davis and Lenat [12]). However, TEIRESIAS and most other "knowledge-based" approaches to knowledge base refinement do not deal with problems arising from deleterious interactions of rules. (For a brief discussion contrasting knowledge-based and empirical approaches see (Valtorta [43]).) For additional discussion of these approaches, see (Wilkins and Tan [45]) and (Tong and Franklin [38]).

Finally, the research reported in this paper has emphasized a theoretical approach to knowledge base refinement. Combined restrictions on the topology of the net, on the values of strenghts, and on the values of certainty factors in cases allow efficient algorithms for rule base refinement, as discussed in Section 5. Whether these favorable situations occur in practice remains to be studied in a program of experiments. It is also possible that randomization techniques may be efficacious for rule base refinement in the average case.

Acknowledgements

The earlier work described in this paper was partly supported by the U.S. Air Force Office of Scientific Research under grants AFOSR-81-0221 and AFOSR-83-8205, with Donald Loveland of Duke University as principal investigator. Donald Loveland completed a weaker version of the first proof of NP-Completeness presented in this paper before the author proved the version given here.

Many thanks to Donald Loveland for useful discussions throughout much of the research reported in this paper. Thanks to Derek Sleeman of the University of Aberdeen for correspondence that stimulated the presentation in Section 4, and to Ronan Reilly of the University of Dublin for correspondence on the topic of neural network training. Some aspects of presentation benefited from discussion with the other participants in the knowledge base refinement track of the Sixth International Workshop on Machine Learning, especially Allen Ginsberg,

Pedro Meseguer, and David C. Wilkins. R. Martin Chavez, an anonymous referee, and Laura P. Valtorta also gave valuable suggestions on matters of presentation.

References

1. Andreassen, S., M. Woldbye, B. Falck, and S.K. Andersen. "MUNIN--A Causal Probabilistic Network for the Interpretation of Electromyographic Findings." *Proceedings of the Tenth International Joint Conference on Artificial Intelligence,* Milan, 366-372, 1987.

2. Biswas, G. and T.S. Anand. "Using the Dempster Shafer Scheme in a Diagnostic Expert System Shell." *Proceedings of the Third AAAI Workshop on Uncertainty in AI,* Seattle, 98-105, 1987.

3. Blum, A. and R.L. Rivest. Training a 3-Node Neural Network is NP-Complete. *Proceedings of the 1988 Workshop on Computational Learning Theory (COLT-88),* 9-18.

4. Breuker, J., B. Wielinga, M. van Someren, R. de Hoog, G. Schreiber, P. de Greef, B. Bredeweg, J. Wielemaker, J.-P. Billault, M. Davoodi, S. Hayward. "Model-Driven Knowledge Acquisition: Interpretation Models." Deliverable task A1, ESPRIT Project 1098, and Memo 87, VF Project Knowledge Acquisition in Formal Domains, 1987.

5. Brooks, R. and J. Heiser. "Some Experience on Transferring the MYCIN System to a New Domain." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2, 5 (September 1980), 477-478.

6. Brownston, L., R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5.* Reading, Massachusetts: Addison-Wesley, 1985.

7. Buchanan, B.G. and E.H. Shortliffe. *Rule-Based Expert Systems.* Reading, Massachusetts: Addison-Wesley, 1984 (reprinted with corrections, May 1985).

8. Caruana, R.A. "The Automatic Training of Rule Bases that Use Numerical Uncertainty Representations." *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence* (1987), 198-204.

9. Clancey, W. J. "Heuristic Classification." *Artificial Intelligence,* 27 (1985), 289-350.

10. Cooper, G.F. "Probabilistic Inference Using Belief Networks is NP-Hard." Stanford University Knowledge Systems Laboratory Memo KSL-82-27, May 1987 (revised July 1988).

11. Craw, S. and D. Sleeman. "The Refinement of Dynamic Knowledge Bases by Collecting Evidence." Typescript, University of Aberdeen, 1988.

12. Davis, R. and D.B. Lenat. *Knowledge-Based Systems in Artificial Intelligence.* New York: McGraw-Hill, 1982.

13. Drastal, G.A. and Kulikowski, C.A. "Knowledge-Based Acquisition of Rules for Medical Diagnosis." *Journal of Medical Systems,* 6, 5 (1982), 433-445.

14. Fu, Li-Min. "Truth Maintenance Under Uncertainty." *Proceedings of the Fourth AAAI Workshop on Uncertainty in Artificial Intelligence,* 119-126, 1988.

15. Fu, Li-Min and Bruce Buchanan. "Enhancing Performance of Expert Systems by Automated Discovery of Meta-Rules." *Proceedings of the First Conference on Artificial Intelligence Applications,* Denver, 107-115, 1984.

16. Gallant, S.I. "Connectionist Expert Systems." *Communications of the ACM,* 31, 2 (February 1988), 152-169.

17. Garey, M.R. and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman, 1979.

18. Gashnig, J. "Prospector: An Expert System for Mineral Exploration," in *Introductory Readings in Expert Systems, ed. D. Michie,* Gordon and Breach Science Publishers, New York, 1982.

19. Ginsberg, A. *Automatic Refinement of Expert System Knowledge Bases.* London: Pitman, 1988.

20. Gold, M. "Language Identification in the Limit." *Information and Control,* 10, 447-474 (May 1967).

21. Gordon, J. and E.H. Shortliffe. "A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space." *Artificial Intelligence,* 26 (1985), 323-357.

22. Hafner, D.E. Letter to "ACM Forum." *Communications of the ACM,* 29, 7 (July 1986), 592.

23. Hayward, S.A., B.J. Wielinga, and J.A. Breuker. "Structured Analysis of Knowledge." *International Journal of Man-Machine Studies,* 26 (1987), 487-498.

24. Heckerman, D.E. and E.J. Horwitz. "On the Expressiveness of Rule-based Systems for Reasoning with Uncertainty." *Proceedings of the Sixth National Conference on Artificial Intelligence,* 121-126 (1987).

25. Judd, S. "Complexity of Connectionist Learning with Various Node Functions." Technical Report 87-60, University of Massachusetts at Amherst, July 1987.

26. Judd, S. "Learning in Neural Networks." *Proceedings of the 1988 Workshop on Computational Learning Theory (COLT-88),* 2-8.

27. Megiddo, N. "Towards a Genuinely Polynomial Algorithm for Linear Programming." *SIAM Journal on Computing,* 12, 2 (May 1983), 347-353.

28. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, California: Morgan-Kaufmann, 1988.

29. Politakis, P.G. "Using Empirical Analysis to Refine Expert Systems Knowledge Bases," Ph.D. Dissertation, Department of Computer Science, Rutgers University, October 1982. (Also Laboratory for Computer Science Research Report CBM-TR-113 and published as a monograph by Pitman, London, in 1985.)

30. Rada, R. "Probabilities and Predicates in Knowledge Refinement." *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems* (1984), 123-128.

31. Rada, R. "Gradualness Facilitates Knowledge Refinement." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7, 5 (September 1985), 523-530.

32. Ruspini, E.H. "Possibility Theory Approaches for Advanced Information Systems." *Computer*, 15, 9 (September 1982), 83-91.

33. Shapiro, E.Y. "Algorithmic Program Debugging." Research Report 237, Yale University, Department of Computer Science (May 1982). (Also published as a monograph with the same title: Cambridge, Massachusetts: MIT Press, 1983.)

34. Shapiro, E.Y. "Logic Programs with Uncertainties: A Tool for Implementing Rule Based Expert Systems." *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)* (1983), 529-532.

35. Sharma, S. and D. Sleeman. "REFINER: A Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement." Typescript, Department of Computer Science, University of Aberdeen.

36. Shortliffe, E.H. *Computer-Based Medical Consultations: MYCIN*. American Elsevier: New York, 1976.

37. Smets, P. "Belief Functions," in *Non-Standard Logics for Automated Reasoning* (P. Smets, E.H. Mamdani, D. Dubois, and H. Prade, Eds.), Academic Press, London, 253-286, 1988.

38. Tong, C. and P. Franklin. "Toward Automated Rational Reconstruction: A Case Study." *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York, 302-307.

39. Valtorta, M. "Knowledge Refinement in Rule Bases for Expert Systems: An Application-Driven Approach." *Proceedings of the First International Workshop on Expert Database Systems*, October 1984.

40. Valtorta, M. "Automating Rule Strengths in Expert Systems." Ph.D. Dissertation, Department of Computer Science, Duke University, April 1987. (Also: Technical Report CS-1987-15, Department of Computer Science, Duke University, and available as Number ADG87-25869 from University Microfilms International.)

41. Valtorta, M. "Automating Rule Strengths in Expert Systems," *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI-88)*, August 1988.

42. Valtorta, M. "Some Results on Knowledge-base Refinement with an Oracle." Technical Report TR89005, Department of Computer Science, University of South Carolina, Columbia, 1989.

43. Valtorta, M. "Knowledge-base Refinement: A Bibliography." Technical Report TR89006, Department of Computer Science, University of South Carolina, Columbia, 1989.

44. Valtorta, M. "The Synthesis of Initial Probability Assignments in Belief Nets is NP-Hard." Technical Report TR89007, Department of Computer Science, University of South Carolina, Columbia, 1989.

45. Wilkins, D.C. and K. Tan "Knowledge Base Refinement as Improving an Incorrect, Inconsistent and Incomplete Domain Theory." *Proceedings of the Sixth International Workshop on Machine Learning*, Ithaca, New York, 332-337.

46. Wilkins, D.C. and B.G. Buchanan. "On Debugging Rule Sets When Reasoning Under Uncertainty." *Proceedings of AAAI-86*, 448-454.