Refinement of Uncertain Rule Bases via Reduction

Charles X.F. Ling* Marco Valtorta†

Abstract

Refining *deep* (multilayer) rule bases of an expert system with uncertainty to cover a set of new examples can be very difficult (NP-hard). We analyze *refinement via reduction*, an approach first proposed in [Ginsberg, 1988b], where it is claimed that this approach eases the complexity of refining rule bases *without uncertainty*. We outline a model of rule bases *with uncertainty*, and give necessary and sufficient conditions on uncertainty combination functions that permit reduction from deep to *flat* (non-chaining) rule bases. We prove that reduction cannot be performed with most commonly used uncertainty combination functions. However, we show that there is a class of reducible rule bases in which the strength refinement problem is NP-hard in the deep rule base, reduction is polynomial, and the flat rule base can be refined in polynomial time. This result also allows polynomial refinement of practical expert systems in the form of rule deletion. Thus, our results provide some theoretical evidence that refinement via reduction is feasible.

Key Words: Refinement in rule bases with uncertainty, inductive learning from examples, automatic knowledge acquisition, complexity analysis.

*Department of Computer Science, University of Western Ontario, London, Ontario, Canada N6A 5B7. E-mail: ling@csd.uwo.ca. Phone: 519-661-3341 (Office). The work has been supported partially by an NSERC Operating Grant.

†Department of Computer Science, University of South Carolina, Columbia, SC 29208, U.S.A. E-mail: mgv@usceast.cs.scarolina.edu. Phone: 803-777-4641 (Office). The beginning of this research was supported by a University of South Carolina summer grant, which is gratefully acknowledged.

1 Introduction

It is well known that building and maintaining large rule bases is a time consuming, error prone "bottleneck" process. Machine learning, a young and exciting field in AI, has been providing promising solutions to the bottleneck problem [Buchanan, 1989]. Learning by induction, the central topic of machine learning, studies how a theory is constructed and revised from data. If the theory is incomplete or imperfect, some of its behavior (data or cases it explains) would be incorrect. The goal of automatic knowledge acquisition is to construct a knowledge base to have the desired behaviors, and to modify or update the knowledge base from unexpected or incorrect behaviors. Inductive learning techniques are useful in automatic knowledge base acquisition [Buchanan, 1989, Bareiss et al., 1989, Valtorta, 1991a, Ling et al., 1993]. In this paper, we will consider the important special case in which the knowledge base (to use the term prevailing in expert systems research) or theory (to use the term prevailing in machine learning research) is a rule base with uncertainty, whose format will be describe precisely in Section 2.

The central problems of automatic rule base acquisition are refinement and synthesis. *Rule base refinement* is the process of modifying an existing rule base in a plausible or conservative way so it performs desired behaviors (or derives a set of correct cases, to be defined). In this paper, we assume all cases are noiseless. This may occur, for example, when the cases are part of a set that the knowledge-based system must handle correctly to be *certified* as fit for some purpose. One kind of conservative revision is called *minimal revision*; i.e. revise the rule base as little as possible. Another kind is to revise the strengths of rules only while keeping the structure of the given rule base unchanged. *Rule base synthesis*, on the other hand, is the process of constructing a new rule base from a set of cases. Thus synthesis is a special form of refinement: it is a refinement from an empty rule base. In this paper, we study both refinement and synthesis problems.

It was proved [Valtorta, 1989, Valtorta, 1991b] that many refinement and synthesis problems in simple rule bases are NP-hard. These rule bases are "simple" in the sense that they are very shallow (but not flat) and have only a small number of intermediate terms. See the references and Section 4 for a more precise description. These worst case results are pessimistic, and they raise serious concerns: if algorithmic refinement and synthesis are infeasible in simple rule bases, is automatic knowledge acquisition possible for practical rule

bases?

Ginsberg in his papers [Ginsberg, 1988b, Ginsberg, 1989] explored *refinement via reduction*, an approach that might ease the refinement problem. A rule base may be represented in a *deep* (multilayer) structure or in a *flat* (non-chaining) one (cf Section 2). The process of transferring a rule base from a deep representation to a flat one is called *reduction*. Briefly, refinement via reduction consists of three steps: reduction, refinement in the reduced (flat) theory, and retranslation. However, Ginsberg's work does not deal with uncertainty. In addition, his refinement algorithm in the flat theory is highly heuristic in an attempt to maintain minimal revision, an NP-hard problem. We view refinement simply as revision of rule base under certain constraint (such as strength refinement, see later) to satisfy all given cases. Under this theoretical framework, refinement without uncertainty can be done trivially without even going through reduction[1]. The reduction process itself is trivial, too[2]. We will show that in a rule base with uncertainty, the trivial refinement algorithm does not work, and reduction cannot always be performed.

Our paper gives theoretical analyses showing that refinement via reduction indeed eases the refinement of some practical rule bases. First, we present *necessary* and *sufficient* conditions for performing reduction. These conditions indicate that, surprisingly, reduction cannot be performed in most practical rule bases with uncertainty. However, we will show that there is a class of reducible theories for which certain refinement problems that are NP-hard in the "deep" theory can be solved in polynomial time in the corresponding reduced theory. This result also allows polynomial refinement of rule bases in the form of rule deletion, an interesting application of refinement via reduction. Note that our result does *not* mean that an NP-hard problem is in P. Since reduction changes the structure of the theory, the result of the refinement in the reduced theory may not correspond to a solution in the original, deep theory. That is, we did not really solve the refinement problem in deep theories (which is NP-hard) efficiently. All we do is to transfer the problem in one representation (deep theory) into another representation (flat theory), and solve the problem efficiently in the new representation. The distinction between the language of the deep theory and the language

[1]The trivial refinement algorithm first specializes the rule base (by adding conditions to the rules) until no unwanted conclusions can be proved. Then it constructs a new, specific rule for each conclusion that can not be derived from the current rule base.

[2]The procedure is the same as expanding a arbitrary and-or Boolean function to DNF (disjunction of conjunctions of inputs variables.)

of the flat theory is analogous to that between the language of concepts and the language of hypotheses in PAC-learning [Kearns, 1990].

The paper is organized as follows: In Section 2, a model of simple rule bases with uncertainty is outlined. This model is used throughout the paper. Section 3 discusses necessary and sufficient conditions for reduction. Sections 4, 5 and 6 prove results on various refinement problems in deep rule bases and in reduced rule bases. These results show that refinement via reduction indeed eases the refinement of practical rule bases. Section 7 discusses refinement via rule deletion, an application of feasible refinement via reduction. Finally, Section 8 is for related work, and Section 9 for conclusions.

2 The Model of the Rule Bases

We study reduction in the truth functional uncertainty model in this paper. In the *truth functional (extensional) uncertainty* model, the uncertainty of consequences of rules is a function (consisting of uncertainty combination functions) of the uncertainties of premises of rules [Pearl, 1988]. Depending on the choice of uncertainty combination functions, our notion of "uncertainty" can represent degree of belief, probability, importance, or any similar notions. Note, however, that the truth functionality requirement restricts the possible semantics for uncertainty. We maintain that the simplicity and computational advantages of truth-functional systems justify their study, despite their semantical problems. See the first chapter of Pearl's book [Pearl, 1988] for a discussion of the tradeoff between complexity and soundness. See also [Wang and Valtorta, 1992] for a specific example of this tradeoff. To prove a refinement problem is NP-hard or polynomially solvable, a particular set of combination functions is chosen. To show conditions for performing reduction, only *constraints* among combination functions have to be introduced.

2.1 Conjunctive Rule Model with Uncertainty

Throughout the paper (except in Section 7, we assume that each rule in the rule base is in the form of the standard propositional Horn clause with uncertainty:

$$\text{IF condition}_1, \ldots, \text{condition}_n \text{ THEN consequence}_1 \text{ WITH STRENGTH } (r), \qquad (1)$$

where the conditions and the consequence are propositions with uncertainties attached. The constant r in the rule (1) is the *strength* (or uncertainty) of the rule. The uncertainties of the propositions and the rules range in a domain B, such as positive integers, integers from 1 to 100, the set {0, 1}, or the closed interval [0, 1] of the reals. The strength of the rule r determines the uncertainty of the consequence from the uncertainties of the conditions. Notice that uncertainty values of the rules and propositions do not affect the proof process (inference) of propositional logic. In addition, no recursion occurs in the rule base (the same assumption is made in [Ginsberg, 1988b, Ginsberg, 1989]).

Although many expert systems including MYCIN allow both disjunction and conjunction in the condition part of the rules, rules in our model are a *simplified* version of rules in MYCIN. One can, upon satisfaction of certain conditions (see Section 3), transfer rules with disjunction in the conditions to another set of rules with only conjunctive conditions as the ones in our model. Such a process may lead to an exponential growth in the number of conjunctive rules. However, such exponential growth is intrinsic to the original disjunctive rule set and is *not* introduced by reduction from conjunctive rules to "flat" rules.

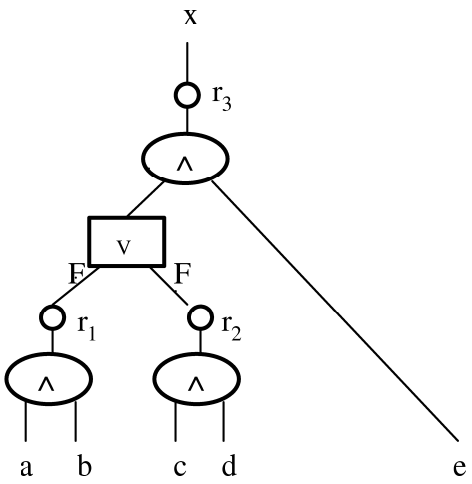
The uncertainties of the conclusions are computed by *uncertainty combination functions* (denoted by $\mathbb{A}$, $\forall$ and $\triangleright$) from the uncertainties of input data and the strengths of the rules. Given a set of true propositions with their uncertainties, new conclusions may be derived using a standard forward chaining inference process. The inference process can be described as follows: each cycle of deduction starts with matching the condition part of each rule (in the form (1) above) with propositions true in the rule base. If all conditions are matched and at least one of them matches the propositions just asserted into the rule base, the rule is fired. The function $\mathbb{A}$, called the *combinator*, is used to combine the uncertainties of the conditions of the rule fired. A frequently used combinator is MIN (the minimum function). The function $\triangleright$, called *carry-over*, then carries the combined uncertainty of the conditions to the consequence of the rule with r (the strength of the rule). A common practice is simply to multiply r with the combined uncertainty of the conditions. If the same consequence is derived from several rules or if the consequence is already in the rule base, its uncertainty is integrated by applying the *integrator* function (denoted as $\forall$) that integrates all uncertainties of the consequence together. The integrator functions frequently used are MAX (the maximum function) and probabilistic sum ($f(x, y) = x + y - xy$). Then all consequences with their uncertainties obtained this way are asserted into the current rule base as new conclusions. Once this cycle of deduction finishes, the next starts from the

current (possibly enlarged) rule base. The process stops if there is no new assertion at the end of a cycle. Since there is no recursion in the theory, the procedure always stops, and all conclusions with their uncertainty values are obtained.

It is convenient to represent the rule base and the inference process in graphs, called *inference nets*. Distinguished nodes are used corresponding to strengths carry-over (circles), combinators (ellipses), and integrators (rectangles). For example, a rule base with three rules:

IF a, b THEN F WITH STRENGTH r_1
IF c, d THEN F WITH STRENGTH r_2
IF F, e THEN x WITH STRENGTH r_3

is represented in Figure 1 (a). Given the uncertainty functions and a case consisting of input propositions with uncertainties, such as $\{(a, 0.8)(b, 0.3), (c, 1), (d, 0.7), (e, 0.9)\}$, the conclusions ($F$ and x) with their uncertainties can be calculated.



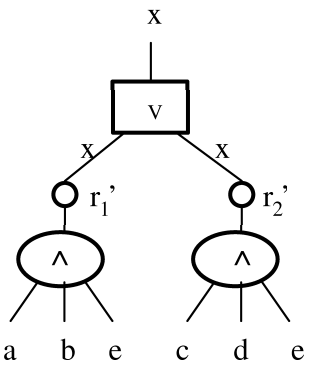


Figure 1: (a) The inference net with three rules. (b) The corresponding reduced net

2.2 Deep vs. Flat Rule Bases

The set of propositions used in the rule bases can be partitioned into 3 subsets: the *input propositions*, the *output propositions*, and the *intermediate propositions*. Both input and output propositions are also called *observable terms*, since their truth values (and uncertain-

ties) can be observed, obtained, or verified. The intermediate propositions are also known as *theoretical terms*. They act as intermediaries and are merely used to derive other observable terms.

We say that a rule base with intermediate propositions has a *deep structure*, and that a rule base without intermediate propositions has a *flat* or *reduced structure*. Such flat rule bases, also called *stimulus-response* production systems, consist of rules asserting conclusions (output proposition) from observations (input propositions) directly, without using intermediate propositions (see Figure 1 (b)). Although stimulus-response production systems have the simplest kind of knowledge base representation, most of the more traditional machine learning work (e.g., [Michalski, 1983]) has focused on these systems.

2.3 The Refinement and Synthesis Problems

First, we define the notion of case and what it means for a theory to derive or cover a case. When a theory derives a case, we say that the case is satisfied in the theory.

Definitions. A *case* of a rule base is the set of input propositions and output propositions with uncertainties attached to them. That is, a case is a set of $\{(p_1, \#p_1), (p_2, \#p_2),$ ..., $(c_1, \#c_1), (c_2, \#c_2), ...\}$, in which p_i are input propositions, c_i are output propositions, and $\#x$ or $\#(x)$ represents the uncertainty of x. The set may be interpreted as "there is a case whose inputs are p_1 with uncertainty $\#p_1$, p_2 with uncertainty $\#p_2$, etc. and whose conclusions are c_1 with uncertainty $\#c_1$, c_2 with uncertainty $\#c_2$ etc." A case is *correct* if it is true in the domain.

The rule base (or the inference net for the whole rule base) realizes a function from vectors of input uncertainties to vectors of output uncertainties. We use $\{(c'_1, \#c'_1), (c'_2, \#c'_2), ...\}$ to denote the actual output vector calculated from a given inference net and the input vector $\{(p_1, \#p_1), (p_2, \#p_2), ...\}$ of the case.

Definitions. A case is *satisfied* in the rule base (or a theory *derives* or *covers* a case) if $\{(c_1, \#c_1), (c_2, \#c_2), ...\} = \{(c'_1, \#c'_1), (c'_2, \#c'_2), ...\}$. Given a set of correct cases S, the *synthesis problem* is to construct a rule base (theory) that derives all cases in S. The *refinement problem* is to modify a given theory T into a theory T' such that T' derives S, and certain features of T is preserved. *Strength refinement* is a refinement in which only rule

strengths may be modified; i.e., the rule structure is preserved.

Three remarks are in order. First, when we discuss the complexity of refinement, the problem size includes the size of the original theory plus the size of all given cases. Second, the refined or synthesized strengths are not constrained. In practice, the new strengths should be close to the old ones according to a suitable metric. This is a version of minimal refinement that is often mentioned in the literature on knowledge base refinement. For example, see the discussion on conservativeness and radicality of a refinement in Chapter 1 of Ginsberg's book [Ginsberg, 1988a]. However, without a formal description of conservativeness in refinement from the given rule bases, the complexity of rule base synthesis is a lower bound on the complexity of refinement[3]. Third, we do not consider incremental refinement. We assume that all cases are available at the same time. This may be impractical in many situations. However, our lower bound time complexity results are not affected, since presentation of cases in order (one at a time or in small batches) is a special case of presentation of cases in a set, and therefore no incremental algorithm is faster than the best batch algorithm. We readily acknowledge that the last two issues deserve further work that is outside the scope of this paper.

3 Necessary and Sufficient Conditions for Reduction

Theory reduction produces from a set of rules a functionally equivalent set without intermediate propositions. More specifically, it transfers a rule base in the deep structure to a functionally equivalent one in the flat structure. *Functional equivalence* means that the deductive closures of the output propositions with their uncertainties are identical. That is, given any set of input propositions, the same set of conclusions is drawn from both theories, and for each conclusion drawn, its uncertainty is the same in both theories.

When represented in classical logic (without uncertainty), rule base synthesis can be trivially simple: just construct a rule for each given case. Theory reduction is also very simple since only the truth of the conclusions needs to be preserved. It is well known that reduction can always be performed in the classical axiomatizable theories. However, we

[3]Problems AER (Approximate Epsilon Refinement) and RSN (Rule Strength Synthesis, No-Switch Case) in [Valtorta, 1991b] are examples of formal descriptions of conservativeness in rule bases.

will show this is not true for theories *with uncertainty*. In this section, we study uncertainty models that are *closed* under reduction. That is, we study necessary and sufficient conditions of uncertainty functions under which intermediate propositions can be eliminated from the theory while the same results in conclusions can be obtained.

The actual algorithm performing reduction (with or without uncertainties) is the same and quite simple: each intermediate proposition in the condition part of any rule is repeatedly replaced by disjunction of conditions of rules which conclude it. The resulting "nested" rules contain no intermediate propositions, and are then "flattened" to disjunctive normal form (reduced rule base). The uncertainty values of the reduced rules have to be calculated during the reduction process. However, as we will see, the uncertainty combination functions must satisfy certain conditions to insure equivalent transformation in rule bases with uncertainty.

The above algorithm performs reduction in polynomial time if the number of intermediate terms is fixed. Otherwise reduction may result in a reduced theory of exponential size. However, such a situation may not be realistic in practice. The size of the problem should capture the size of the input variables as the number of observable features or symptoms as well as the size of the cases given, but not the intermediate propositions since they are "theoretical".

3.1 General Conditions

Uncertainties of all propositions and rules should lie in a linearly ordered set B. For example, B might be the interval of real numbers $[0, 1]$, or the natural numbers with the usual order, or the set {unlikely, possible, likely, certain} with the given order showing decreasing uncertainty. We assume B is closed under the uncertainty functions, and that B has an upper bound $\mathbf{1}$, representing absolute certain knowledge. A rule can be stated without doubt if the strength of a rule is $\mathbf{1}$. In this case, we assume the uncertainty of the condition part is directly carried over to the consequence, i.e. $x \triangleright \mathbf{1} = x$ for all x in B.

It is also reasonable to expect that the order of computing uncertainties by $\wedge$ and $\vee$ does not matter. Thus $\wedge$ and $\vee$ are associative and commutative; furthermore, we need only consider $\wedge$, $\vee$ and $\triangleright$ to be binary operations. As usual, we assign priorities to the three operators so that $\wedge$ has highest priority, followed by $\triangleright$ and then $\vee$. Thus we may omit parentheses when there is no ambiguity. In writing equations involving uncertainty

values, we will use proposition names (such as a) directly rather than "the uncertainty of a" or $\#a$) if there is no ambiguity. Without loss of generality, we assume that after reduction no two rules have the same conditions and conclusion because two such rules can always be merged into a single one with a new strength.

3.2 Preserving Uncertainty Values

Recall that to preserve conclusions and their uncertainty values in reduction given any case, we require the following:

- The same set C of conclusions is drawn from each theory.
- For each $c \in C$, the uncertainty of c is the same in either theory.

Example: Assume the original rules are (also see Figure 1 (a)):

If a, b then F (r_1)

If c, d then F (r_2)

If F, e then x (r_3)

To preserve the deducibility of x from the original set of rules given any assertion of observational terms (i.e. a, b, c, d or e), the rules in the reduced theory must be (also see Figure 1 (b)):

If a, b, e then x (r_1')

If c, d, e then x (r_2').

We seek conditions on uncertainty functions such that r_1' and r_2' can be determined and the uncertainty of x in the reduced theory will be the same as in the original theory. That is: if all of a, b, c, d, and e are asserted, then:

$$\#(x) = ((a \mathbin{\&} b \mathbin{\triangleright} r_1) \mathbin{\vee} (c \mathbin{\&} d \mathbin{\triangleright} r_2)) \mathbin{\&} e \mathbin{\triangleright} r_3,$$

while in the reduced theory:

$$\#(x) = (a \mathbin{\&} b \mathbin{\&} e \mathbin{\triangleright} r_1') \mathbin{\vee} (c \mathbin{\&} d \mathbin{\&} e \mathbin{\triangleright} r_2').$$

Therefore,

$$((a \mathbin{\&} b \mathbin{\triangleright} r_1) \mathbin{\vee} (c \mathbin{\&} d \mathbin{\triangleright} r_2)) \mathbin{\&} e \mathbin{\triangleright} r_3 = (a \mathbin{\&} b \mathbin{\&} e \mathbin{\triangleright} r_1') \mathbin{\vee} (c \mathbin{\&} d \mathbin{\&} e \mathbin{\triangleright} r_2')$$

Similarly, when a, b, and e are asserted, and when c, d, and e are asserted, we have:

$$(a \mathbin{\&} b \mathbin{\&} e \mathbin{\triangleright} r_1') = (a \mathbin{\&} b \mathbin{\triangleright} r_1) \mathbin{\&} e \mathbin{\triangleright} r_3$$

$$(c \mathbin{\&} d \mathbin{\&} e \mathbin{\triangleright} r_2') = (c \mathbin{\&} d \mathbin{\triangleright} r_1) \mathbin{\&} e \mathbin{\triangleright} r_3.$$

(To be continued)

The following theorem give necessary conditions on $\mathbb{A}$, $\mathbb{V}$ and $\triangleright$ so that the reduced theory will preserve uncertainty values.

Theorem 3.1 *The three conditions below are necessary and sufficient conditions for reduction with uncertainty*
1. $x \mathbb{A} (y \mathbb{V} z) = (x \mathbb{A} y) \mathbb{V} (x \mathbb{A} z)$.
2. $(x \mathbb{V} y) \triangleright r = (x \triangleright r) \mathbb{V} (y \triangleright r)$.
3. $(x \triangleright r) \mathbb{A} y = (x \mathbb{A} y) \triangleright f(r)$, *where* f *is a function determined by* $\mathbb{A}$ *and* $\triangleright$.

Proof:
To prove that the first condition is necessary, consider the following simple rule base:

If a then F $(\mathbf{1})$
If b then F $(\mathbf{1})$
If F, c then x $(\mathbf{1})$

To preserve the conclusion x (independently of uncertainty value), the set of rules in the reduced theory is of the form:

If a, c then x (r_1)
If b, c then x (r_2).

Now consider preservation of uncertainty values. When all of a, b, and c are asserted, we have $(a \mathbb{V} b) \mathbb{A} c = (a \mathbb{A} c) \triangleright r_1 \mathbb{V} (b \mathbb{A} c) \triangleright r_2$; when a, c are asserted, we have $(a \mathbb{A} c) = (a \mathbb{A} c) \triangleright r_1$; and when b, c are asserted, we have $(b \mathbb{A} c) = (b \mathbb{A} c) \triangleright r_2$. Thus, for all a, b, and c, we find

$$(a \mathbb{V} b) \mathbb{A} c = (a \mathbb{A} c) \mathbb{V} (b \mathbb{A} c).$$

To prove that the second condition is necessary, consider the following simple rule base:

If a then F $(\mathbf{1})$
If b then F $(\mathbf{1})$
If F then x (r)

In the reduced theory, there are two rules:

If a then x (r_1)
If b then x (r_2).

It follows that $(a \mathbb{V} b) \triangleright r = a \triangleright r_1 \mathbb{V} b \triangleright r_2$ when a, b are asserted; $a \triangleright r = a \triangleright r_1$ when a alone is asserted; and $b \triangleright r = b \triangleright r_2$ when b alone is asserted. Thus we conclude:

$$(a \mathbb{V} b) \triangleright r = (a \triangleright r) \mathbb{V} (b \triangleright r).$$

Notice that we cannot necessarily conclude $r_1 = r_2 = r$.

To prove that the third condition is necessary, consider the following simple rule base:

If a then F (r_1)

If F, c then x $(\mathbf{1})$

In the reduced theory, there is only one rule:

If a, c then x (r').

When a is asserted, we have

$(a \blacktriangleright r_1) \mathbin{\&} c = a \mathbin{\&} c \blacktriangleright r'$, where r' depends only on r_1.

Although these three conditions are derived from three small and specific rule bases, we now prove that they are actually *sufficient* for reducing general rules with uncertainties. Recall that we are assuming commutativity and associativity for $\&$ and $\vee$. The first two conditions enable us to "flatten" any propositional formula to disjunctive normal form, and the third condition enables us to pull out $\blacktriangleright$ from the middle of propositions to the strength part of the rule, in both cases preserving uncertainty values. □

Example. (continued from the beginning of this Section) When the three conditions are met, the $\#(x)$ of the original theory can be rewritten as follows:

$$\begin{aligned}
\#(x) &= ((a \mathbin{\&} b \blacktriangleright r_1) \vee (c \mathbin{\&} d \blacktriangleright r_2)) \mathbin{\&} e \blacktriangleright r_3 \\
&= ((a \mathbin{\&} b \blacktriangleright r_1) \mathbin{\&} e \vee (c \mathbin{\&} d \blacktriangleright r_2) \mathbin{\&} e) \blacktriangleright r_3 \\
&= (a \mathbin{\&} b \blacktriangleright r_1) \mathbin{\&} e \blacktriangleright r_3 \vee (c \mathbin{\&} d \blacktriangleright r_2) \mathbin{\&} e \blacktriangleright r_3 \\
&= (a \mathbin{\&} b \mathbin{\&} e) \blacktriangleright r_1' \blacktriangleright r_3 \vee (c \mathbin{\&} d \mathbin{\&} e) \blacktriangleright r_2' \blacktriangleright r_3
\end{aligned}$$

where $r_1' = f(r_1)$ and $r_2' = f(r_2)$, for f the function of theorem 3. If f is the identity function $(f(x) = x)$, then the two rules in the reduced theory are (as shown in Figure 1 (b)):

If a, b, e then x $(r_1 \blacktriangleright r_3)$

If c, d, e then x $(r_2 \blacktriangleright r_3)$. (End of the example)

There are natural functions $\&$, $\vee$, and $\blacktriangleright$ which satisfy all three conditions: for example, we may take both $\&$ and $\blacktriangleright$ to be the minimum function, and $\vee$ to be the maximum function; or both $\&$ and $\blacktriangleright$ to be multiplication, and $\vee$ to be plus (with the domain of natural numbers with an upper bound "infinity"); or $\&$ to be MIN, $\blacktriangleright$ to be the multiplication function, $\vee$ to be the maximum function and uncertainty ranges on $\{0, 1\}$. In these cases (as in many others) the function f of theorem 3 is just $f(r) = r$. Functions satisfying conditions for reduction are adopted in several rule based systems. Some expert

systems, such as Prospector [Gashnig, 1981, Duda et al., 1976] and AL/X [Reiter, 1980], use the "fuzzy" formulae for conjunction (MIN) and disjunction (MAX) [Quinlan, 1983]. The same is true for some fuzzy control systems and rule based systems based on fuzzy logic (although not all of them). For example, [Togai and Watanabe, 1986]. Note that in all these cases, the uncertainty values must be either 0 or 1; otherwise (if they are real numbers from 0 to 1) the third condition of theorem 3.1 will not be satisfied. However, a slight extension of our model with uncertainty in [0, 1] allows us to refine a rule base through rule deletion. See Section 7 for details.

We have also obtained a set of sufficient and necessary conditions for reduction when only the *rank order* of uncertainties of conclusions or the *most likely* conclusion is preserved. These conditions are slightly relaxed in comparison with the ones for preserving uncertainty values shown in this section. See [Ling and Dawes, 1990] for detailed discussions and proofs.

3.3 What If a Theory Cannot be Reduced?

Surprisingly, the MYCIN type of uncertainty combination functions are not compatible with reduction. (We consider here a simplified version of the original MYCIN combination function, as described in [Shortliffe, 1976], as opposed to the one described in [Buchanan and Shortliffe, 1984].) In MYCIN-type rule based expert systems, the certainty factor (CF) is a real number in $[0, 1]$. The particular $\mathbin{\&}$ used is the minimum function, $\rhd$ is the multiplication function, and $\vee(x, y) = x \oplus b = x + y - xy$. It is easy to verify that this choice of $\mathbin{\&}$, $\vee$ and $\rhd$ does not satisfy condition 1 of Theorem 3.1, even when rule strengths are limited to {0, 1}. As a simple example, look at the following MYCIN rule base which contains three rules with strengths r_1, r_2, r_3, from {0, 1}:

If a then F (r_1)
If b then F (r_2)
If F then x (r_3),

The uncertainty of x is

$(ar_1 + br_2 - ar_1br_2)r_3$.

On the other hand, the reduced rule would take the form

If a then x (r'_1)
If b then x (r'_2)

The uncertainty of x would be

$ar'_1 + br'_2 - ar'_1 br'_2$.

If the rule base were reducible, we would be able to find constants r'_1 and r'_2 such that

$(ar_1 + br_2 - ar_1 br_2)r_3 = ar'_1 + br'_2 - ar'_1 br'_2$.

That is, the uncertainty value of conclusion x would be preserved for any given uncertainty values of input a, b and any given strengths r_1, r_2, r_3. It is easy to verify that such constants r'_1 and r'_2 do not exist. Thus, this rule base cannot be reduced.

Although the necessary and sufficient constraints discussed in the previous section is for a complete (all intermediate propositions are eliminated) and total (the whole rule base is reduced) reduction, they are applicable on "partial" reduction as well. That is, if these conditions are not met, partial reduction that eliminates some intermediate propositions can not be done; neither can part of the rule base be reduced. Such a rule base has a "rigid" structure which would be difficult to be transferred into another functionally equivalent form. However, inequivalent transfer of the rule base may result in very unexpected behaviors or destroy early prototypes. Thus, it can be inflexible to work with models of expert systems in which the conditions for reduction are not satisfied.

As an example, the refinement problem in the deep rule structure of MYCIN is NP-complete [Valtorta, 1991b]. Our conditions for reduction indicate that it is, in general, impossible to transform MYCIN rules into an equivalent reduced set for a possibly feasible refinement. Moreover, we prove in section 6.2 that such a refinement in the reduced theory is also NP-hard. This provides evidence that maintaining a large expert system like MYCIN by knowledge engineers or knowledge acquisition programs is inherently difficult in the worst case. In practice, it may be possible to use only a rough estimation of probabilistic sum[4], or to perform approximate refinements that do not require an absolute consistency with a given set of cases.

4 Strength Refinement in Deep Rule Bases

In this section, we list many strength refinement problems that are NP-complete when commonly used uncertainty combination functions are chosen. Some of these problems will be shown to be polynomially solvable after reduction. The topology of the inference net used in

[4]For example, only the largest two values are integrated.

the first two theorems below is a tree, with one layer of two intermediate propositions. The in-degree of combinators from input propositions is two, the in-degree of the two integrators for intermediate propositions may grow with the size of the inputs. That is, there is no limit on number of rules concluding the two intermediate propositions. The combination functions are chosen among many popularly used functions. For example, the combinator and integrator are MIN and MAX respectively, the rule strengths are multiplicative; and the uncertainty range B is the set {0,1} or the reals from 0 to 1 inclusive ([0,1]). See Figure 2(a). The strength refinement problem considered here starts with a rule base with a correct structure but with *no known strengths*. Given a set of cases, strengths of all rules are to be determined so the resulting rule base covers all given cases with correct uncertainty values or correct rank order of uncertainty values. Because no strength is known, such a strength refinement problem is also called *strength synthesis* problem. Refinement can be as hard as synthesis, when a proper measure of conservativeness is not given.

The following strength synthesis problems are established and proved to be NP-complete [Valtorta, 1991b]. We state the results here without proofs and give some explanation if necessary. For the detailed proofs, refer to [Valtorta, 1991b]. The complexity of the refinement is measured with respect to the size of theory and cases, which, under some technical assumptions, is characterized by the number of input propositions and the number of cases given. Of course the complexity of refinement in such trees is a lower bound of the complexity of arbitrary inference net. For the purpose of proof, it is sufficient to consider the inference net with the topology described in Figure 2(b). This is a special case of the inference net in Figure 2(a) when the combinators are MIN, the strengths of input propositions are $\{x_1, 1, x_2, 1, ...\}$ (i.e. $y_1 = y_2 = ... = 1$), and the carry-over function is multiplicative. The *Restricted Rule Strength Synthesis (RS)* problem is to find[5] an assignment of strengths such that the network, when given the input part of each case, computes the correct output part. If such an assignment exists, we say that the network satisfies the cases for that assignment.

Theorem 4.1 *RS with* $B = \{0, 1\}$ *is NP-complete.*

Note that such rule bases are polynomially reducible (Section 3). As we will show in the

[5]Problems RS and RSP are defined as decision problems in [Valtorta, 1991b]. The corresponding search problems are NP-hard.

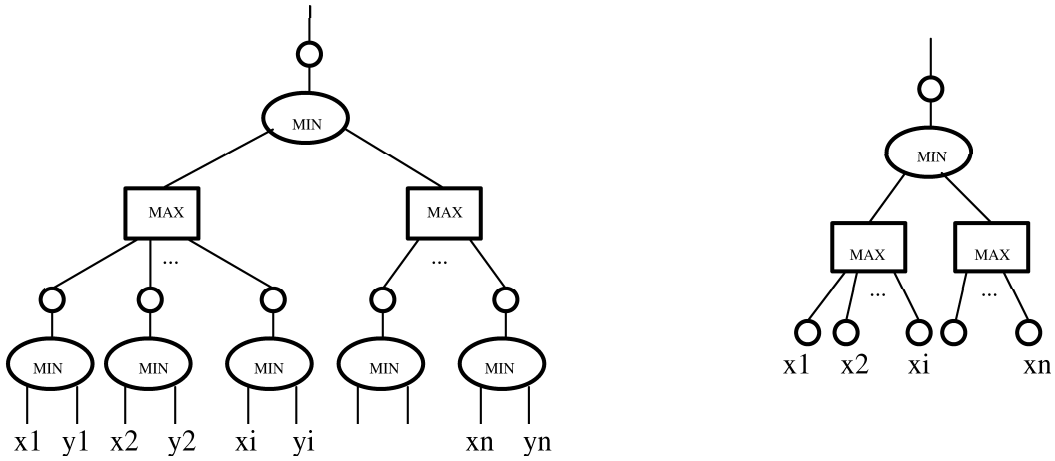


Figure 2: (a) The topology of a deep rule base. (b) The simplification of (a) in the proof.

next section, the reduced rule bases can be solved in polynomial time. The next theorem extends the previous one when the uncertainty ranges in $[0, 1]$.

Theorem 4.2 *RS with* $B = [0, 1]$ *is NP-hard.*

Many expert systems adopt MYCIN-like certainty factor and combination functions. Rule strength synthesis is NP-hard for MYCIN like inference trees where the combinator function is MIN, the integrator function is probabilistic sum ($P+$), and the carry-over is multiplication. The *Restricted Rule Strength Synthesis with MIN/P+ (RSP)* problem is to find an assignment of strengths in [0,1] or $\{0, 1\}$ that satisfies the given cases.

Theorem 4.3 *RSP with* $B = \{0, 1\}$ *or* $B = [0, 1]$ *is NP-complete.*

It has also been shown that if only the rank order of the conclusions ordered by their uncertainties is important, the strength refinement and synthesis problems are still NP-hard [Valtorta, 1991b]. That is, it is NP-hard to synthesize rule strengths in rule bases even when the most likely conclusion, or a list of conclusions ordered by uncertainty values, matters.

5 Strength Refinement in Reduced Rule Bases

Let us consider some of the refinement and synthesis problems in the reduced theory with a flat structure. (Recall that the reduction algorithm is given in Section 3.) First, we will show that some strength refinement problems can be solved in polynomial time, while the same problem is NP-hard in the deep, polynomially reducible rule bases. These results demonstrate that there are cases in which reduction makes the refinement problems strictly easier to solve. Then we will show that some other strength refinement and complete refinement problems in the reduced theory is still NP-hard.

5.1 Strength Refinement with MIN and MAX

Recall that strength refinement in the deep theory is NP-complete, that is, RS with $B = \{0,1\}$ or $B = [0,1]$ is NP-complete. Now we show that the same problem with $B = [0,1]$ is polynomially solvable in the reduced theory using the same combination functions. Assume a set of cases and a rule base with a fixed flat structure is given. For each given case j of total q cases, assume the outputs of the MIN nodes are $p_1^j, p_2^j, ...p_s^j$, and the output of the MAX node is v^j. See Figure 4. The algorithm in Figure 3 processes cases incrementally, and each time weights are lowered just enough to get this case correct. The algorithm will return the setting of all strengths such that the rule base covers all given cases, if such a setting exists. If the uncertainty is allowed to be 0 and 1 only ($B = \{0,1\}$), then change the last line in the above algorithm to "else output $a_1, ..., a_q$ if all of them are 0 or 1 as the results of refinement".

As an example, if there are total of 3 inputs to the integrator MAX ($s = 3$ in Figure 4), and there are two cases with

$p_1 = 0.6,\ p_2 = 0.9,\ p_3 = 0.5;\quad v = 0.4$

$p_1 = 0.7,\ p_2 = 0.8,\ p_3 = 0.9;\quad v = 0.5,$

then the algorithm first calculates the possible values for the strengths $a_1,\ a_2$ and a_3 as:

$a_1 = MIN(0.4/0.6,\ 0.5/0.7) = 2/3$

$a_2 = MIN(0.4/0.9,\ 0.5/0.8) = 4/9$

$a_3 = MIN(0.4/0.5,\ 0.5/0.9) = 5/9$

Then the algorithm verifies if the desired outputs can still be obtained by the strengths just

---

for $1 \leq i \leq s$
  set $a_i = \infty$
for each case j $(1 \leq j \leq q)$
  for $1 \leq i \leq s$ set $a_i = min(v^j/p_i^j, a_i)$
for each case j $(1 \leq j \leq q)$
  verify if $max(a_1 \times p_1^j, ..., a_s \times p_s^j) = v^j$
if the verification step fails for any case
  then refinement fails: no rule strengths exist to cover all cases
  else output $a_1, ..., a_s$ as the result of refinement

Figure 3: Algorithm that refines the strengths of rules in polynomial time

calculated:

$$MAX(0.6 \times (2/3),\ 0.9 \times (4/9),\ 0.5 \times (5/9)) = 0.4$$
$$MAX(0.7 \times (2/3),\ 0.8 \times (4/9),\ 0.9 \times (5/9)) = 0.5$$

Obviously the verification succeeds.

It can easily be shown that the above algorithm is polynomial. Let q be the number of cases and s be the number of strengths. Let the number of input propositions be n. (Note that the number of strengths is equal to the number of MIN nodes and $n = 2s$.) The first two steps of the algorithm (initialization and computation of strengths) takes total time $O(qs)$. The third step of the algorithm (verification) takes total time $O(qs)$. The last step (output) takes time $O(s)$. It is obvious that the algorithm terminates. We claim that all cases are satisfied if and only if the algorithm terminates successfully. Since each case is satisfied if the algorithm terminates successfully, the "if" part is trivial. We show the "only if" part by contradiction. If a strength is set to a larger value than that computed in step 2, there is a case k for which the output of the MAX node is greater than v^k. Conversely, if a strength is set to a smaller value than that computed in step 2, there is a case k for which the output of the MAX node is less than v^k. The preceding discussion can be summarized in the following Theorem.

Theorem 5.1 *The Algorithm in figure 3 refines the strengths of rules in polynomial time.*

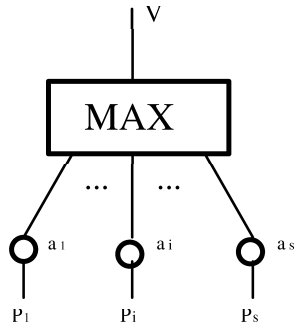


Figure 4: Reduced theory in which refinement is polynomial

The complexity of the algorithm is $O(qn)$ where q is the number of cases in the set, and n is the number of input propositions (which is twice the number of strengths in the inference net).

Here comes the main conclusion of the paper. Consider a deep theory in which the number of inputs (n) and the size of cases (q) may increase as the size of the problem in this model, but the number of intermediate propositions is fixed at 2. As discussed in Section 3, such a theory can be reduced in polynomial time ($O(n^2)$), with an increase of input propositions from n to $O(n^2)$. By Theorem 5.1, the total time complexity of reduction and strength refinement is polynomial ($O(n^2q)$) with respect to the size of the original theory n and q, While by Theorems 4.1 and 4.2, strength refinement in the deep theory is NP-hard. Thus, we have shown that indeed there are cases in which changing the representation of rules via reduction makes the refinement problems *strictly easier* to solve.

Note that restricting strengths of rules to {0,1 } does not mean that we just get propositional logic as in Ginsberg's work, because uncertainties associated to propositions can still range in (e.g.) [0,1]. Refinement with uncertainty is harder than refinement without uncertainty. Restricting strengths to {0,1 } is equivalent to rule deletion or rule selection from a large rule base. In Section 7, we will show an interesting application of rule deletion in a slightly extended model of rule bases that allow arbitrary rule strengths.

5.2 Strength Refinement with Probabilistic Sum

Although we have shown that some strength refinement problems that are NP-complete in the deep theory can be solved efficiently in the flat theory, our next result is negative: the strength refinement problem in the reduced theory is still NP-hard for other integrator functions. A particularly important example is *probabilistic sum (P+),* denoted as $\oplus$ and defined as $a \oplus b = a + b - ab$. Probabilistic sum is used in MYCIN [Shortliffe, 1976] and many other expert systems with certainty factors. The problem of *Rule Strength Synthesis in a Flat Structure with Probabilistic Sum (SRP)* is to find an assignment of strengths that satisfies all given cases. See Figure 5(b). The inference net in Figure 5(b) is a special case of the net in Figure 5(a), when the strengths of input propositions are $p_1, 1, p_2, 1, ...$ This is analogous to the situation described in Figures 2(a) and 2(b).

Figure 5: (a) An inference net. (b) The simplification of (a) used in SRP instances

Theorem 5.2 *SRP is NP-hard.*

Proof: See Appendix.

Our result indicates that strength refinement even in flat rule bases using probabilistic sum as integrator is intractable. Probabilistic sum is used widely in many expert systems including MYCIN. Thus, to facilitate refinement, probably a simpler function (such as MIN or the probabilistic sum of the largest two uncertainty values) may be used.

6 Complete Refinement

Since we have shown that strength refinement in the reduced theory with uncertainty is polynomial, and complete refinement in the rule base without uncertainty is also polynomial (and trivial), one might tend to think that complete refinement (both the structure and the strengths of the rule base can be modified) in the reduced theory with uncertainty is also easy. After all, a reduced theory with uncertainty is the simplest form of rule bases with uncertainty, and it seems that allowing the structure of rules to be modified too gives more freedom in revision and thus eases the complete refinement task. It is surprising that this is not the case: total refinement is intractable. Intuitively, this is because the strengths of disjunctive rules *interact* with each other.

In this section we prove that complete refinement in the flat structure is NP-complete. It is still unknown if complete refinement (with arbitrary change of structure) in the deep theory is NP-hard, or harder than synthesis with no change in structure, although we strongly conjecture that the answer to both questions is yes. At least we know that complete refinement in a deep theory that does not create over a *fixed* number of intermediate terms in addition to those in the original deep theory is NP-hard. For otherwise we could apply such algorithm on a flat theory and reduce the revised theory in polynomial time (since there is at most a fixed number of intermediate terms). This contradicts the proof (given below) that complete refinement in the flat theory is NP-complete.

Figure 6: (a) An inference net in the flat structure. (b) A modification of the net

The flat inference network has MAX as the root node, a layer of strengths, and a layer of MIN nodes (cf. Figure 6(a)). We now allow for changes in the structure of the net of

the following form: we can vary the number of inputs to each MIN node and the overall number of MIN nodes. This is equivalent to allowing the addition of new rules, the deletion of old rules, and the modification of rules by deletion and addition of propositions in their premises. Note that the overall number of inputs is fixed, since it is the size of the input part of a case. Figure 6(b) shows a modified net that satisfies our restriction on change of structure.

We call the synthesis problem just described CSRT (*Complete Synthesis in Reduced Theory*) where changes in structure are allowed and prove that the new synthesis problem is NP-hard. An interpretation of this result is that complete synthesis in the flat structure is, at least in some cases, strictly harder than strength synthesis in the flat network of fixed structure (if $P \neq NP$). This result confirms a conjecture contained in [Valtorta, 1991b].

Theorem 6.1 *CSRT with* $B = [0, 1]$ *is NP-hard.*

Proof: See Appendix.

7 Refinement through Rule Deletion

We have found a class of polynomial reducible theories where strength refinement is intractable in the deep theory (the RS problem in Section 4), and polynomial in the corresponding reduced theory (Section 5). The class of rule bases uses MIN as the combinator, MAX as the integrator, and strengths as 0 or 1 only. It seems that this class of rule bases is quite restrictive. Is this result useful in practical expert system refinement?

Strength refinement where strengths can only be assigned to 0 or 1 accounts for rule *addition* and *deletion*. It has been argued [Ma and Wilkins, 1991] that strengths of rules should not be modified ("tweaked") anyway, and refinement should only allow the deletion of rules. In other words, if we have a large set of rules for various situations, the refinement of rule bases is reduced to *selecting* a proper subset of the rules for some specific situation. Thus, rule deletion is interesting and important in practice.

Restricting the strengths of rules to be 0 or 1 only does not mean that rules cannot have uncertainty. In fact, since the uncertainties of rules cannot be modified, they can be

incorporated as a component of the rules. That is, the uncertainty of the conditions of rules can be altered by the carry-over function ($\triangleright$) before being integrated by the combinator ($\wedge$). For example, any conjunctive rule in the form:

IF a, b then x (r)

can be translated into

IF a (r), b (r) then x (1)

as long as

$(a \wedge b) \triangleright r = (a \triangleright r) \wedge (b \triangleright r)$.

This constraint is satisfied when strength is multiplicative (i.e. the $\triangleright$ is multiplication). In fact, MYCIN allows alteration of uncertainty of conditions in rules using *predicate functions* just like this.

Thus, to refine via rule deletion a given set of such rules with rule strengths ranging in $[0, 1]$, we first push all rule strengths into the condition part of the rules, leaving all the rules to have strengths of 1. Then the rule base is reduced in polynomial time. Refinement in the form of rule deletion can be done in polynomial time in the reduced rule base using the algorithm discussed in Section 5. Notice that it is proved to be NP-hard to do rule deletion if the rule base is not reduced. Since many rule bases [Gashnig, 1981, Duda et al., 1976, Reiter, 1980, Togai and Watanabe, 1986] do use MIN, MAX, and [0,1], our method provides a feasible refinement of the rule bases in the form of rule deletion and rule selection.

8 Related Work

Knowledge base refinement is a form of inductive learning from examples. For summaries of classic work on inductive learning, see [Angluin and Smith, 1983] [Dietterich and Michalski, 1983]. Much work on refinement has been done. For a bibliography, see [Valtorta, 1991a]. In contrast to the present study, most of the previous research avoids numerical uncertain representations, and uses heuristics without average or worse case complexity analyses. It seems that almost no work devoted to refinement in uncertain rule bases from a complexity theory perspective has yet been published.

Ginsberg [Ginsberg, 1988b, Ginsberg, 1989] first suggested that refinement via reduction might be feasible and robust. (For other research on refinement via reduction see [Jackson, 1991,

Zlatareva, 1992].) However, his work does not deal with uncertainty, and it employs many heuristics. For example, his refinement algorithm in the reduced theory uses five procedures [Ginsberg, 1988b]: massive label generalization and specialization, focused label generalization and specialization, and patching. Each of the five procedures is highly heuristic. One experiment was reported, but there is no analysis comparing the time complexity of refinement in the reduced theory with the one in the original theory and their predictive power. It was unclear if refinement via reduction indeed helps in solving these problems.

Theory reduction is a pre-process of refinement. Methodologically, reduction is a form of compilation: It is similar to the operationalization process of Explanation Based Learning (EBL) [Mitchell et al., 1986]. The operational criterion in EBL corresponds to the observational (non-theoretical) criterion in reduction. The difference between reduction and EBL is that, first, the compilation is not the purpose but just a pre-process in refinement; second, reduction is performed completely in the whole rule base without using any positive examples; and third, EBL does not deal with uncertainty. Theory reduction with uncertainty is much more complicated than the operationalization process in EBL, since both conclusions and their uncertainty values (or rank order) need to be preserved.

Solving a problem by first changing its representation is a common and very important approach in AI. Some early work on problem reformulation includes [Amarel, 1982, Korf, 1980]. However, very little of the work done in the area (such as papers in [Benjamin, 1990]) seems directly applicable to reduction in knowledge base refinement.

9 Conclusions

We have demonstrated that rule base refinement to cover a set of given cases is computationally intractable (NP-complete) in deep rule bases that use truth functional (extensional) uncertainty models[6]. We have shown that some of these refinement problems become feasible if they are solved in rule bases with a flat structure obtained via reduction, while other refinement problems remain NP-complete. We have provided constraints on truth functional uncertainty models that permit reduction. We have also provided theoretical evidence, for

[6]Some analysis of refinement in non-truth functional (intensional) probabilistic network models has been done in [Valtorta, 1990, Valtorta and Loveland, 1992].

the first time, supporting the effectiveness of refinement via reduction when reduction is possible.

Our work may provide some guidelines for future work in automatic knowledge acquisition and expert system construction. First of all, unless carefully chosen, uncertainty combination functions make refinement computationally infeasible, not only in a deep rule structure, but also in a flat one. Probabilistic sum, for example, may be too complicated to use for integration of uncertainty. Second, unless carefully chosen, uncertainty combination functions may not satisfy the constraints for reduction. This implies that the rule structure of the expert system is "rigid": it is no longer an easy task to change the rule structure while maintaining the equivalence of the rule base. Inequivalent transferring of the rule base may destroy the prototype built earlier, resulting in unexpected behaviors. In addition, it is not possible to transfer the rule base into the flat structure in which refinement is possibly easier to solve. Third, a total refinement (modify the rule base to fit *all* given cases) may be too crude a standard as the criterion for successful refinement, especially if the training data are noisy. Although some approximate refinement problems have been studied [Valtorta, 1991b], more future work is needed in this area. Fourth, knowledge acquisition via cases (or examples) *only* may be insufficient for efficient learning. In automatic knowledge acquisition, other assistance besides cases and examples, such as explanations of the example (or hints as in [Angluin, 1987]), selected typical and "good" examples[7], queries (e.g. membership of instances or generic cases), knowledge base support tools [Musen, 1989], and interactions with human experts, may be required, justifying a shift of activity from automatic to *computer-assisted* knowledge acquisition.

Acknowledgments

The authors wish to thank Mike Dawes for many discussions on conditions for performing reduction, Don Loveland for many general and technical discussions on knowledge base refinement, Ray Mooney for discussions and for sharing some unpublished work on the RAPTURE system with us, Ying Hua for drawing many figures in the paper, and the anonymous referees for numerous suggestions that led to an improved paper.

[7]For example, the use of selected examples to improve the efficiency of model inference of first order logic is discussed in [Ling, 1991].

10 Appendix

Theorem 5.2:
The problem of Rule Strength Synthesis in a Flat Structure with Probabilistic Sum (SRP) is NP-hard.

Proof:
One in Three Satisfiability (OTS) [Garey and Johnson, 1979] (page 259) will be transformed into SRP. The variant in which no clause in the formula contains a negative literal will be used. The generic OTS instance is a formula in 3-conjunctive normal form, with no negated variables. The question is whether there is a model for the expression such that each clause has exactly one true variable.

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 SRP such that the Question has answer yes if and only if E has a model in which only one variable per clause is true.

Let n be the number of distinct propositional 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.) (Name the variables $x_1, ..., x_n$ for convenience.) The number of leaves in the inference tree of the corresponding SRP instance is n. The number of cases in the corresponding SRP instance is $2m$.

There are 2 cases for each clause in E. The cases are defined as follows. Let a and b be a pair of numbers such that $0 < a < b < 1$. Let a (generic) clause contain the variables x_i, x_j, x_k. The input part of the first case for each clause has $p_i = p_j = p_k = a$ and 0 everywhere else. The output part of the first case for each clause is a. To obtain the second case for this clause, substitute b for a. The reader can easily verify that the algorithm just given runs in time polynomial in the size of E. As an example, Figure 7 shows the instance of SRP corresponding to $E = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$. In the Figure, T_1 and T_2 correspond to the first clause in E, while T_3 and T_4 correspond to the second clause in E.

The "if part" is simpler. If variable x_i in the model for E is true, set s_i to 1. Otherwise, set it to 0. This insures that exactly one of the uncertainties corresponding to each case is 1 and the other two are 0. Therefore, each case is satisfied.

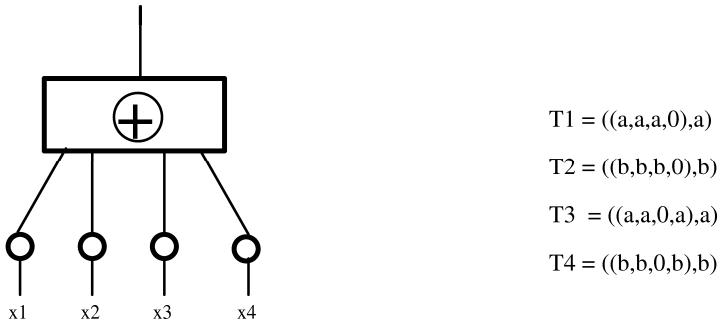


Figure 7: Instance of SRP corresponding to $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$.

The "only if" part is proved now. Assume that we have a yes-instance of SRP. It will be shown that, in order for an instance of SRP to be a yes-instance, it must be that exactly one of the s_i corresponding to each case is 1 and the other two are 0. By assigning true to variable corresponding to this single s_i, a model for E is obtained that satisfies the "one in three" condition. Consider 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 s_i corresponding to the cases is 1 and the other two are 0. Call the strengths x, y, and z. The pair of cases is satisfied if and only if the following system has a solution:

$$ax \oplus ay \oplus az = a$$
$$bx \oplus by \oplus bz = b,$$

i.e., after carrying out the probabilistic sums (indicated by $\oplus$) and dividing each side by a,

$$x + y - axy + z - axz - ayz + a^2xyz = 1$$
$$x + y - bxy + z - bxz - byz + b^2xyz = 1.$$

If any two of x, y, and z 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 and divide by $(b-a)$, we obtain:

$$xy + xz + yz = (b + a)xyz.$$

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. Thus

$$xy + xz + yz > 2xyz > (b + a)xyz$$

Therefore it is impossible that

$$xy + xz + yz = (b + a)xyz.$$

It has been shown that SRP is NP-hard. In order to complete the proof that SRP is NP-complete, it remains to show that SRP is in NP. A non-deterministic program to solve SRP has a loop whose body assigns (non-deterministically) a value to each uncertainty and tests whether for that assignment the function realized by the inference net satisfies all cases. Since the test can be performed in deterministic polynomial time, the whole program runs in non-deterministic polynomial time. □

Remark: We have not specified the possible values for $s_1, ..., s_n$ in the statement of problem SRP. The proof of NP-hardness shows that SRP is NP-complete even when the values are restricted to be in 0,1. If there are more than a constant number of different values, SRP remains NP-hard, but we cannot show it to be in NP. Similarly, we have not specified the possible values for the input part of the cases. The proof shows that SRP is NP-hard even when the values are restricted to a and b, $0 < a < b < 1$. Finally, SRP is NP-complete even if both input and s_i values are restricted as just outlined at the same time.

Theorem 6.1:
The problem of Complete Synthesis in Reduced Theory (CSRT) with $B = [0, 1]$ is NP-hard.

Proof:
We transform monotone three-conjunctive normal form satisfiability (MSAT) [Garey and Johnson, 1979] to CSRT. The generic MSAT instance has the form $E = c_1 \wedge c_2 \wedge ... \wedge c_n$, with n distinct variables $x_1, x_2, ..., x_n$. (Rename variables, if necessary.) Each clause contains only three un-negated variables (in which case it is a *positive clause*) or negated variables (in which case it is a *negative clause*). The question is whether there is a satisfying truth assignment (i.e., a model) for the expression (i.e., formula) E.

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

The CSRT instance has $2n+2$ inputs and $(2n+1)+(2n+3)+m = 4n+m+4$ cases. All the cases have output .4, except two, as will be noted in due course. For mnemonic reasons that will become apparent later, input $n+1$ is called the *POS* input; input $2n+2$ is called

the *NEG* input; input $n+1+i$ is the *complementary* input of input i.

We call each of the first $2n+1$ cases a *variable case*. Each pair of the first $2n$ cases corresponds to a variable in E. For each variable x_i, the first case has input i set to .4, input POS set to .9, input $n+1+i$ set to .4, input NEG set to 0, and all other inputs set to 1. The second case has input i set to .4, input POS set to 0, input $n+1+i$ set to .4, input NEG set to .9, and all other inputs set to 1. The $(2n+1)^{st}$ case has output 0, inputs POS and NEG set to 0, and all other inputs set to 1.

Each of the $2n+3$ cases in the second group is a *strength case*. The i^{th} case in this group has .4 in position i ($1 \leq i \leq 2n+2$) and 1 everywhere else, except for the last case that has all inputs and the output set to 1. This is the only case with output set to 1.

Each of the m cases in the third group is a *clause case*. Case j corresponding to clause $c_j = (x_{j1}, x_{j2}, x_{j3})$ is built as follows when c_j is a positive clause: inputs j_1, j_2, j_3 are set to .4, input $n+1$ is set to 0. All other inputs are set to 1. When c_j is a negative clause, inputs $n+1+j_1, n+1+j_2$, $n+1+j_3$ are set to .4, input $n+1$ is set to 0, and all other inputs are set to 1.

Clearly these cases can be built in time polynomial in n and m. Therefore this construction takes polynomial time in the size of a reasonable encoding of E.

Example: Instance of CSRT corresponding to $E = (x_1 \vee x_2 \vee x_3) \wedge (-x_1 \vee -x_2 \vee -x_4)$.

$T_1 = ((.4,1,1,1,.9,.4,1,1,1,0),.4)$
$T_2 = ((.4,1,1,1,1,.4,1,1,1,.9),.4)$
....
....
....
$T_7 = ((1,1,1,.4,.9,1,1,1,.4,0),.4)$
$T_8 = ((1,1,1,.4,0,1,1,1,.4,.9),.4)$
$T_9 = ((1,1,1,1,0,1,1,1,1,0),0)$
$T_{10} = ((.4,1,1,1,1,1,1,1,1,1),.4)$
....
....
$T_{19} = ((1,1,1,1,1,1,1,1,1,1,.4),.4)$
$T_{20} = ((1,1,1,1,1,1,1,1,1,1,1),1)$

$T_{21} = ((.4, .4, .4, 1, .9, 1, 1, 1, 1, 0), .4)$
$T_{22} = ((1, 1, 1, 1, 0, .4, .4, 1, .4, .9), .4)$

Each case has $2n + 2 = 10$ inputs and one output. Cases T_1 through T_9 ($9 = 2n + 1$) are variable cases. Cases T_{10} through T_{20} are strength cases. (Note that there are 11 such cases and $2n + 3 = 11$.) Cases T_{21} and T_{22} are clause cases. (Note that there are two clauses in E.) (End of example.)

We now show that the instance of CSRT built according to the algorithm just given is a yes-instance if and only if E is satisfiable. We start by proving two lemmas.

Lemma 10.1 *All strengths in an instance of CSRT that satisfies the strength cases have value 1.*

Proof: It is easy to verify that all strength cases are satisfied by setting all strengths to 1. We need to show that if at least one strength is less than one, at least one strength case is not satisfied. First observe that the last strength case (in the example, T_{19}) cannot be satisfied if all strengths are less than 1. Assume that there is a strength (say s) at the output of a MIN node with input x_i and possibly other inputs (cf. Figure 8), and $s \leq 1$.

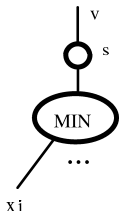


Figure 8: The output of one MIN node is less than 0.4.

Consider the i^{th} strength case. For this case, since $x_i = .4$ and $s = 1$, the output of the MIN node (say v) is less than .4. Recall in our nets that all outputs of MIN nodes are input to a MAX node (cf. Figures 6(a) and (b)). Therefore, in order for the output of the net to be .4, it must be that none of the other MIN nodes have output greater than .4 and at least one has output equal to .4. Since all the other inputs in the i^{th} case have value 1, this requires that the strengths of all other MIN nodes be at most .4. We have shown that, if

one strength has value less than 1, then all strengths must have value less than 1. But, as we have already observed, the last strength case cannot be satisfied if all strengths are less than 1. Contradiction! □

Lemma 10.2 *The i^{th} pair of variable cases is satisfied if and only if*
either *the i^{th} input is MINned with the $(n+1)^{st}$ input and the $(n+1+i)^{th}$ input is MINned with the $(2n+2)^{nd}$ input,*
or *the $(n+1+i)^{th}$ input is MINned with the $(n+1)^{st}$ input and the i^{th} input is MINned with the $(2n+2)^{nd}$.*

Proof: The last variable case requires that each input must be MINned with either POS, or NEG, or both. Consider the j^{th} pair of variable cases. If input j is MINned with POS (NEG), it is clear that the complementary input $n+1+j$ cannot be MINned with POS (NEG). But each input must be MINned with either POS or NEG. Therefore, the complementary input must be MINned with either NEG or POS. □

Note that Lemmas 10.1 and 10.2 require any *solution* of CSRT, i.e., every inference tree such that all cases are satisfied, to have the form shown in Figure 9. All inputs are divided in two groups, one assigned to the same node as POS (the *POS node)*, the other assigned to the same node as NEG (the *NEG node).*

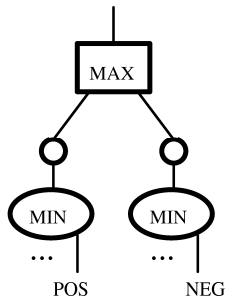


Figure 9: The structure of a solution of CSRT (see text)

Lemma 10.3 *If E is satisfied then (the corresponding instance of) CSRT is a yes-instance.*

Proof: We give an algorithm to construct a solution of CSRT, i.e., an inference tree such that all cases are satisfied. In the solution, all strengths are one. (Therefore all strength

cases are satisfied, by Lemma 10.1.) The topology of the solution is given in Figure 9. Note that there are only two MIN nodes. The *POS node* has POS as one of its inputs. The *NEG node* has NEG as one of its inputs. Since E is satisfiable, it has one or more models. Choose a model. If x_{ij} is true (false) in the model, let input i_j be input to the POS (NEG) node and input n+1+i be input to the NEG (POS) node.

We have already shown that all strength cases are satisfied. Lemma 10.2 allows to conclude that each variable case is satisfied since each pair of inputs corresponding to the same variable is assigned to a different node.

Since at least one of the literals in a clause is true in the model, at least one of the inputs with value .4 in the model is assigned to the POS (NEG) node for a positive (negative) clause. Therefore, each clause case is satisfied. □

Lemma 10.4 *If CSRT is a yes-instance, than E is satisfied.*

Proof: We give an algorithm to construct a model of E from the CSRT instance.

If input i is assigned to the POS (NEG) node and input $n+1+i$ is assigned to the NEG (POS) node, then let x_i be true (false) in the model. The other variables are assigned true or false arbitrarily.

We now show that this algorithm indeed constructs a model. First of all, Lemma 10.2 guarantees that it is impossible for the algorithm to assign true and false to the same variable, therefore the algorithm builds an interpretation. To show that it is a satisfying interpretation (i.e., a model), consider first the generic positive clause in E. The generic positive clause (say, C) in E is satisfied if (at least) one of the variables in it is true, or, equivalently, not all of the variables are false. Since pairs of complementary variables are assigned to pairs (POS, NEG) or (NEG, POS) by Lemma 10.1, by the algorithm just given, all the variables in C would be false only if the inputs corresponding to them were assigned to to the NEG node and the complementary inputs were assigned to the POS node. Clearly, such an assignment would violate the corresponding clause case, contradicting the fact that CSRT is a yes-instance. Similarly for negative clauses. □

Based on Lemmas 10.3 and 10.4, the main Theorem 6.1 is proved. □

References

[Amarel, 1982] Amarel, S. (1982). Expert behavior and problem representations. In Elithorn, A. and Banerji, R., editors, *Artificial and Human Intelligence*. North-Holland, New York.

[Angluin, 1987] Angluin, D. (1987). Learning propositional horn sentences with hints. Technical Report RR-590, Department of Computer Science, Yale University.

[Angluin and Smith, 1983] Angluin, D. and Smith, C. (1983). A survey of inductive inference: theory and methods. *Comput. Surveys*, 15:237–269.

[Bareiss et al., 1989] Bareiss, R., Porter, B. W., and Murray, K. S. (1989). Supporting start-to-finish development of knowledge bases. *Machine Learning*, 4(3/4).

[Benjamin, 1990] Benjamin, D. (1990). *Change of Representation and Inductive Bias*. Kluwer Academic, Norwell, MA.

[Buchanan, 1989] Buchanan, B. G. (1989). Can machine learning offer anything to expert systems? *Machine Learning*, 4(3/4).

[Buchanan and Shortliffe, 1984] Buchanan, B. G. and E. H. Shortliffe (1984) *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA.

[Dietterich and Michalski, 1983] Dietterich, T. and Michalski, R. (1983). A comparative review of selected methods from learning from examples. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine learning: An artificial intelligence approach*. Morgan Kaufmann Publishers.

[Duda et al., 1976] Duda, R., Hart, P., and Nilsson, N. (1976). Subjective bayesian methods for rule-based inference systems. Technical Report 124, Artificial Intelligence Center, SRI International.

[Garey and Johnson, 1979] Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman.

[Gashnig, 1981] Gashnig, J. (1981). Prospector: an expert system for mineral exploration. In Bond, A., editor, *State of the Art Report on Machine Intelligence*. Pergamon Infotech, London.

[Ginsberg, 1988a] Ginsberg, A. (1988a). *Automatic Refinement of Expert System Knowledge Bases.* Pitman Publishing, London.

[Ginsberg, 1988b] Ginsberg, A. (1988b). Knowledge-base reduction: a new approach to checking knowledge bases for inconsistency and redundancy. In *Proceedings of AAAI-88.*

[Ginsberg, 1989] Ginsberg, A. (1989). Knowledge base refinement and theory revision. In *Proceedings of the Sixth International Workshop of Machine Learning.*

[Jackson, 1991] Jackson, P. (1991). Prime implicates: their computation and use. In *Proceedings of the 11th International Workshop on Expert Systems and their Applications*, pages 53 – 64.

[Kearns, 1990] Kearns, M. (1990). *The Computational Complexity of Machine Learning.* The MIT Press, Cambridge, MA.

[Korf, 1980] Korf, R. (1980). Toward a model of representation changes. *Artificial Intelligence*, 14:41–78.

[Lapointe et al., 1993] Lapointe, S., Ling, X., and Matwin, S. (1993). Constructive inductive logic programming. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence (IJCAI-93)*, pages 1030–1036. Morgan Kaufmann Publishers.

[Ling, 1991] Ling, X. (1991). Inductive learning from good examples. In *Proceedings of Twelfth International Conference on Artificial Intelligence (IJCAI-91)*, pages 751–756. Morgan Kaufmann Publishers.

[Ling et al., 1993] Ling, X., Cherwenka, S., and Marinov, M. (1993). A symbolic model for learning the past-tenses of English verbs. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence (IJCAI-93)*, pages 1143–1149. Morgan Kaufmann Publishers.

[Ling and Dawes, 1990] Ling, X. and Dawes, M. (1990). Horn-clause theory reduction with preservation of uncertainty rank. In *Proceedings of International Symposium on Computational Intelligence 90.*

[Ma and Wilkins, 1991] Ma, Y. and Wilkins, D. (1991). Improving the performance of inconsistent knowledge bases via combined optimization method. In *Proceedings The Eighth International Workshop on Machine Learning (ML-91).* Morgan Kaufmann.

[Michalski, 1983] Michalski, R. (1983). A theory and methodology of inductive learning. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine learning: An artificial intelligence approach*. Morgan Kaufmann Publishers.

[Mitchell et al., 1986] Mitchell, T., Keller, R., and Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80.

[Musen, 1989] Musen, M. A. (1989). Automated support for building and extending expert models. *Machine Learning*, 4(3/4).

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

[Quinlan, 1983] Quinlan, J. (1983). Inferno: A cautious approach to uncertain inference. *The Computer Journal*, 26(2):255–269.

[Reiter, 1980] Reiter, J. (1980). *AL/X: An Expert System Using Plausible Inference*. Intelligent Terminals Ltd., Oxford, UK.

[Shortliffe, 1976] Shortliffe, E. (1976). *Computer based medical consultations: MYCIN*. Elsevier, Amsterdam.

[Togai and Watanabe, 1986] Togai, M. and Watanabe, H. (1986). Expert system on a chip: An engine for real-time approximate reasoning. *IEEE Expert*, 1(3):55–62.

[Valtorta, 1989] Valtorta, M. (1989). Some results on knowledge base refinement with an oracle. Technical Report TR89005, Department of Computer Science, University of South Carolina.

[Valtorta, 1990] Valtorta, M. (1990). More results on the complexity of knowledge base refinement: Belief networks. In *Proceedings of the Seventh International Conference of Machine Learning*, pages 419–426.

[Valtorta, 1991a] Valtorta, M. (1991a). Knowledge base refinement: A bibliography. *Journal of Applied Intelligence*, 1(1):87 – 94.

[Valtorta, 1991b] Valtorta, M. (1991b). Some results on the computational complexity of refining confidence factors. *International Journal of Approximate Reasoning*, 5(2).

[Valtorta and Loveland, 1992] Valtorta, M. and Loveland, D. (1992). On the complexity of belief network synthesis and refinement. *International Journal of Approximate Reasoning*, 7(3–4).

[Wang and Valtorta, 1992] Wang, S. and Valtorta, M. (1992). On the conversion of rule bases into belief networks. *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing*, pages 363–368.

[Zlatareva, 1992] Zlatareva, N. (1992). Truth maintenance systems and their application for verifying expert system knowledge bases. *Artificial Intelligence Review*, 6(1):67 – 110.