To appear in the
Proceedings of the Twelfth International Conference on Machine Learning (MLC-95),
Lake Tahoe, July 1995.

The Challenge of Revising an Impure Theory

Russell Greiner
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540-6632
Email: greiner@scr.siemens.com *Phone:* (609) 734-3627

Abstract

A *pure* rule-based program will return a set of answers to each query; and will return the same answer set even if its rules are re-ordered. However, an *impure* program, which includes the Prolog `cut` "!" and `not(·)` operators, can return different answers if the rules are re-ordered. There are also many reasoning systems that return only the *first* answer found for each query; these first answers, too, depend on the rule order, even in *pure* rule-based systems. A theory revision algorithm, seeking a revised rule-base whose *expected accuracy*, over the distribution of queries, is optimal, should therefore consider modifying the order of the rules. This paper first shows that a polynomial number of training "labeled queries" (each a query coupled with its correct answer) provides the distribution information necessary to identify the optimal ordering. It then proves, however, that the task of determining which ordering is optimal, once given this information, is intractable even in trivial situations; *e.g.*, even if each query is an atomic literal, we are seeking only a "perfect" theory, and the rule base is propositional. We also prove that this task is not even approximable: Unless $P = NP$, no polynomial time algorithm can produce an ordering of an n-rule theory whose accuracy is within n^γ of optimal, for some $\gamma > 0$. We also prove similar hardness, and non-approximatability, results for the related tasks of determining, in these impure contexts, (1) the optimal *ordering of the antecedents*; (2) the optimal set of *rules to add* or (3) to *delete*; and (4) the optimal *priority values for a set of defaults.*

1 Introduction

A knowledge-based system (*e.g.*, an expert system, logic program or production system) will return incorrect answers if its underlying knowledge base (a.k.a. its "theory") contains incorrect or mis-organized information. A "theory revision" process attempts to convert such faulty theories into more accurate ones — *i.e.*, theories whose answers correspond more closely to the real world. Many such processes work by hill-climbing in the space of theories, using as operators simple theory-to-theory transformations, such as adding or deleting a rule, or adding or deleting an antecedent within a rule. Another class of transformations *rearrange* the order of the rules, or of the antecedents. These transformations will effectively modify the performance of any knowledge-based system written in a shell that uses operators corresponding to PROLOG's `cut` "!" or `not(·)`, as well as any system that returns only the *first* answer found; this class of shells includes TESTBENCH[1] and other fault-hierarchy systems, and prioritized default theories [Gro91], as well as PROLOG [CM81].

The goal of a theory revision process is to improve the accuracy of the reasoning system on its performance task of answering queries. Section 2 first defines this objective more precisely: as identifying the revision (*i.e.*, "sequence of transformations") that produces a theory whose expected accuracy, over a given distribution of queries, is maximal. It also proves that a polynomial number of training samples (each a specific query paired with its correct answer) is sufficient to provide the information needed to identify a revision whose accuracy is arbitrarily close to optimal, with arbitrarily high probability. Section 3 then presents our main results, showing first that this task is intractable[2] even in trivial situations — *e.g.*, even if each query is

[1]TESTBENCH is a trademark of Carnegie Group, Inc.

[2]Throughout, we will assume that $P \neq NP$ [GJ79], which implies that any NP-hard problem is intractable. This also implies certain approximation claims, presented below.

an atomic literal, we are only seeking a "perfect" ordering (which returns the correct answer to each given query), and the knowledge base is propositional. It also proves intractable the task of finding the smallest number of "individual re-orderings" required required to produce a perfect ordering. We then prove that this task is also non-approximable; *i.e.*, unless $P = NP$, no poly-time algorithm can identify an ordering of an n-rule theory whose accuracy is within n^γ of optimal, for some $\gamma > 0$. This section also proves similar hardness, and non-approximatability, results for the related tasks of determining the optimal ordering of the rule antecedents, and the optimal set of rules to add (resp., delete) in the impure case. Section 4 then quickly overviews reasoning using a "prioritized default theory", and proves that the above results hold, *mutatis mutandis*, in this context as well. Section 5.1 presents several generalizations of our framework. The appendix includes proof sketches of many of the theorems; the complete set of proofs appear in the extended paper [Gre95a].

We first close this introduction by describing some related research.

Related Research: This paper describes the complexity of a particular form of *theory revision*. While there are many implemented theory revision systems (including AUDREY [WP93], FONTE [MB88], EITHER [OM94] and Δ [LDRG94]), most deal (in essence) with the "pure" Horn clause framework, seeking all answers to each query; they therefore do not consider the particular class of transformations described in this paper. The companion paper [Gre95b] analyses the classes of transformations used by those other systems: adding or deleting either a rule or an antecedent within a rule, in the standard *pure* context. Among other results, it proves that the task of finding the optimal set of new rules to add (resp., existing rules to delete) is intractable, but can be approximated to within a factor of 2, in this context.

Second, Bergadeno *et al.* [BGT93] note that learning *impure* logic programs, which include the cut operator, is more difficult than learning pure programs; our paper gives additional teeth to this claim, by showing a particular task (*viz.*, learning the best set of rules to add or to delete) that can be trivially approximated in the context of pure programs, but which is not approximatable for impure programs — see especially Theorem 8.

Third, Valtorta [Val89, Val90, LV91] also considers the computational complexity of modifying a theory. Those papers, however, deal with a different type of modifications: *viz.*, adjusting the numeric "weights" within a given network (*e.g.*, altering the certainty factors associated with the rules), but not changing the structure by arranging rules or antecedents. Wilkins and Ma [WM94] show the intractability of determining the best set of rules to *delete* in the context of such weighted rules, where a conclusion is believed if a specified function of the weights of the supporting rules exceeds a threshold. Our results show that this "optimal deletion" task is not just intractable, but is in fact, non-approximatable, even in the (impure) propositional case, when all rules have unit weight and a single successful rule is sufficient to establish a conclusion.

Finally, this paper has some superficial similarities with [Gre91], as both articles consider the complexity of (in essence) ordering a set of rules. However, while [Gre91] deals with the *efficiency* of finding *any* answer to a given query, this paper deals with the *accuracy* of the particular answer returned.

2 Framework

Section 2.1 first describes propositional PROLOG programs; Section 2.2 then extends this description to predicate calculus. Section 2.3 discusses the sample complexity of the theory revision process.

2.1 Propositional Horn Theories

We define a "theory" as a ordered list of Horn clauses (a.k.a. "rules"), where each clause includes at most one positive literal (the "head") and an ordered list of zero or more literal antecedents (the "body"). A theory is considered "impure" if it includes any rule whose antecedents use either the PROLOG cut "`!`" or negation-as-failure "`not(·)`" operator. See [CM81] for a description of how PROLOG answers queries in general, and in particular, how it uses these operators. The two most relevant points, here, are that PROLOG processes the theory's rules, and each rule's antecedents, in a particular order; and on reaching a cut antecedent, within a rule of the form "σ `:-` τ_1`, ... , !, ... ,` τ_n`.`", PROLOG will not consider any of the other rules whose heads unify with σ.

As a trivial example, consider the theory

$$\mathrm{T}_1 \quad = \quad \left\{ \begin{array}{l} \texttt{q :- !, fail.} \\ \texttt{q.} \\ \texttt{r :- not(q).} \end{array} \right\} \qquad (1)$$

Given the query "`q`", PROLOG first finds the rules whose respective heads unify with this goal (which are the first two), and processes them in the order shown (top to bottom). On reaching the "`!`" antecedent in the "`q :- !, fail.`" rule, PROLOG will commit to this rule, meaning it will now not consider the subsequent atomic rule "`q.`". PROLOG will then try to prove the "`fail`" subgoal, which will fail as T_1 contains no rules whose head unifies with this subgoal. This causes the top-level "`q`" query to fail as well. Now consider the "`r`" query, and notice that it will succeed here as "`q`" had failed; in general, `not(` τ `)` succeeds whenever its argument τ fails, and fails whenever τ succeeds.

Now let T_2 be the theory that differs from T_1 only be exchanging the order of the first two clauses; *i.e.*,

$$\mathrm{T}_2 = \left\{ \begin{array}{l} \texttt{q.} \\ \texttt{q :- !, fail.} \\ \texttt{r :- not(q).} \end{array} \right\} . \qquad (2)$$

Here, the `q` query will succeed, and so the `r` query will fail.

Borrowing from [Lev84, DP91], we also view a theory T as a function that maps each query to its proposed answer; hence, $\mathrm{T}: \mathcal{Q} \mapsto \mathcal{A}$, where $\mathcal{Q}$ is a (possibly infinite) set of queries, and $\mathcal{A} = \{ \texttt{No}, \texttt{Yes} \}$ is the set of possible answers. Hence, given the T_1 and T_2 theories defined above, $\mathrm{T}_1(\texttt{q}) = \texttt{No}$, $\mathrm{T}_1(\texttt{r}) = \texttt{Yes}$, and $\mathrm{T}_2(\texttt{q}) = \texttt{Yes}$, $\mathrm{T}_2(\texttt{r}) = \texttt{No}$.

For now, we will assume that there is a single correct answer to each question, and represent it using the real-world oracle $\mathcal{O}: \mathcal{Q} \mapsto \mathcal{A}$. Here, perhaps, $\mathcal{O}(\texttt{q}) = \texttt{No}$, meaning that "q" should not hold.

Our goal is to find a theory that is as close to $\mathcal{O}(\cdot)$ as possible. To quantify this, we first define the "accuracy function" $a(\cdot, \cdot)$ where $a(\mathrm{T}, \sigma)$ is the accuracy of the answer that the theory T returns for the query σ (implicitly wrt the oracle $\mathcal{O}$):

$$a(\mathrm{T}, \sigma) \quad \overset{def}{=} \quad \begin{cases} 1 & \text{if } \mathrm{T}(\sigma) = \mathcal{O}(\sigma) \\ 0 & \text{otherwise} \end{cases}$$

Hence, as $\mathcal{O}(\texttt{q}) = \texttt{No}$, $a(\mathrm{T}_1, \text{"q"}) = 1$ as T_1 provides the correct answer while $a(\mathrm{T}_2, \text{"q"}) = 0$ as T_2 returns the wrong answer.

This $a(\mathrm{T}, \cdot)$ function measures T's accuracy for a single query. In general, our theories must deal with a range of queries. We model this using a stationary probability function $Pr: \mathcal{Q} \mapsto [0, 1]$, where $Pr(\sigma)$ is the probability that the query σ will be posed. Given this distribution, we can compute the "expected accuracy" of a theory, T:

$$\mathrm{A}(\,\mathrm{T}\,) \;=\; E[\,a(\mathrm{T}, \sigma)\,] \;=\; \sum_{\sigma \in \mathcal{Q}} Pr(\sigma) \times a(\mathrm{T}, \sigma) \;.$$

We will consider various sets of possible theories, $\Sigma(\mathrm{T}) = \{\mathrm{T}_i\}$, where each such $\Sigma(\mathrm{T})$ contains the set of theories formed by applying various transformations to a given theory T; for example, $\Sigma^{OR}(\mathrm{T})$ contains the $n!$ theories formed by rearranging the clauses in the n-clause theory $\mathrm{T} = \langle \varphi_i \rangle_{i=1}^n$. Our task is to identify the theory $\mathrm{T}_{opt} \in \Sigma(T)$ whose expected accuracy is maximal; *i.e.*,

$$\forall\, \mathrm{T}' \in \Sigma(\mathrm{T}):\; \mathrm{A}(\,\mathrm{T}_{opt}\,) \;\geq\; \mathrm{A}(\,\mathrm{T}'\,) \;. \qquad (3)$$

There are two challenges to finding such optimal theories. The first is based on the observation that the expected accuracy of a theory depends on the distribution of queries, which means different theories will be optimal for different distributions. While this distribution is not known initially, it can be estimated by observing a set of samples (each a query/answer pair), drawn from that distribution. Section 2.3 below discusses the number of samples required to obtain the information needed to identify a good $\mathrm{T}^* \in \Sigma(\mathrm{T})$, with high probability.

We are then left with the challenge of computing the best theory, once given this distributional estimate. Section 3 addresses the computational complexity of this process, showing that the task is not just intractable,[3] but it is not even approximatable — *i.e.*, no efficient algorithm can even find a theory whose expected accuracy is even close (in a sense defined below) to the optimal value.

2.2 Predicate Calculus

To handle predicate calculus expressions, we consider answers of the form[4] $\texttt{Yes}[\{X_i/\texttt{v}_i\}]$, where the expression within the brackets is a binding list of the free variables, corresponding to the *first* answer found to the query. For example, given the theory

$$\mathrm{T}_{pc} = \left\{ \begin{array}{l} \texttt{tall(john). rich(fred). rich(john).} \\ \texttt{eligible(X) :- rich(X), tall(X).} \end{array} \right\}$$

(where the ordering is the obvious left-to-right, top-to-bottom traversal of these clauses), the query `tall(Y)` will return

$$\mathrm{T}_{pc}(\texttt{tall(Y)}) = \texttt{Yes}[\{Y/\texttt{john}\}] \;;$$

the query `rich(Z)` will return the answer

$$\mathrm{T}_{pc}(\texttt{rich(Z)}) = \texttt{Yes}[\{\,Z/\texttt{fred}\,\}]$$

(recall the system returns only the first answer it finds); and

$$\mathrm{T}_{pc}(\texttt{eligible(A)}) = \{\ \texttt{Yes}[\{A/\texttt{john}\}]\ \}$$

(here the system had to backtrack).

2.3 Sample Complexity

We use following standard Computational Learning Theory theorem to bound the number of samples required to obtain the information needed to identify a good $\mathrm{T}^* \in \Sigma(\mathrm{T})$ with high probability; showing in particular how this depends on the space of theories $\Sigma(\mathrm{T})$ being searched:

[3] As $a(\mathrm{T}, q)$ requires computing $\mathrm{T}(q)$, which can require proving an arbitrary theorem, this computation alone can be computationally intractable, if not undecidable. Our results show that the task of finding the optimal theory is intractable *even given a poly-time oracle for these arbitrary derivations.* Of course, as we are considering only Horn theories, these computations are guaranteed to be poly-time in the propositional case [BCH90].

[4] Following PROLOG's conventions, we will capitalize each variable, as in the "X_i" above.

Theorem 1 (from [Vap82, Theorem 6.2])
Given a class of theories $\Sigma = \Sigma(T)$ *and constants* $\epsilon, \delta > 0$, *let* $\mathrm{T}_* \in \Sigma$ *be the theory with the largest empirical accuracy after*

$$M_{upper}(\Sigma, \epsilon, \delta) \quad = \quad \left\lceil \frac{2}{\epsilon^2} \ln\left(\frac{|\Sigma|}{\delta}\right) \right\rceil$$

samples (each a labeled query), drawn from the stationary distribution. Then, with probability at least $1-\delta$, *the expected accuracy of* T_* *will be within* ϵ *of the optimal theory in* Σ*; i.e., using the* T_{opt} *from Equation 3,* $Pr[\, A(\,\mathrm{T}_*\,) \geq A(\,\mathrm{T}_{opt}\,) - \epsilon\,] \geq 1-\delta$.

This means a polynomial number of samples is sufficient to identify an ϵ-good theory from Σ with probability at least $1-\delta$, whenever $\ln(|\Sigma|)$ is polynomial in the relevant parameters. Notice this is true for $\Sigma = \Sigma^{OR}(\mathrm{T})$: Using Stirling's Formula, $\ln(|\Sigma^{OR}(\mathrm{T})|) = O(n \ln(n))$, which is polynomial in the size of the initial theory $n = |\mathrm{T}|$. We will see that (a variant of) this "$\ln(|\Sigma|) = \mathrm{poly}(|\mathrm{T}|)$" claim is true for every class of theories Σ considered in this paper.

3 Computational Complexity

Our basic challenge is to produce a theory T_{opt} whose accuracy is as large as possible. As mentioned above, the first step is to obtain enough labeled samples to guarantee, with high probability, that the expected accuracy of the theory whose empirical accuracy is largest, T_*, will be within ϵ of this T_{opt}'s. This section discusses the computational challenge of determining this T_*, given these samples. It considers four different classes of theories:

- $\Sigma^{OR}(\mathrm{T})$ (resp., $\Sigma^{OA}(\mathrm{T})$, $\Sigma^{AR}(\mathrm{T})$ and $\Sigma^{DR}(\mathrm{T})$) is the set of theories formed by re-ordering the clauses of a given initial theory T (resp., re-ordering the antecedents of T's clauses, adding new clauses to T, and deleting existing clauses from T).

Notice each $\Sigma \in \{\, \Sigma^{OR},\ \Sigma^{OA},\ \Sigma^{AR},\ \Sigma^{DR}\,\}$ is a function mapping a theory to a set of theories.

To state the task formally: For any theory–to–set-of-theories mapping Σ,

Definition 1 ($\mathrm{DP}(\,\Sigma\,)$ **Decision Problem)**
INSTANCE:
- *Initial theory* T*;*
- *Labeled training sample* $S = \{\langle q_i, \mathcal{O}(\,q_i\,)\rangle\}$ *containing a set of queries and the correct answers; and*
- *Probability value* $p \in [0,1]$.

QUESTION: *Is there a theory* $\mathrm{T}' \in \Sigma(\mathrm{T})$ *such that* $A(\,\mathrm{T}'\,) = \frac{1}{|S|}\sum_{\langle q_i, \mathcal{O}(\,q_i\,)\rangle \in S} a(\mathrm{T}', q_i) \ \geq\ p$ *?*

(Notice we are simplifying our notation by writing $\mathrm{A}(\,\mathrm{T}'\,)$ for the approximation to $\mathrm{A}(\,\mathrm{T}'\,)$ based on the training sample S.)

We will also consider the following special cases:

- $\mathrm{DP}_{Perf}(\,\Sigma\,)$ requires that $p=1$;
  i.e., seeking perfect theories, rather than "optimal" theories $\mathrm{DP}_{Opt}(\,\Sigma\,)$;
- $\mathrm{DP}_{Pur}(\,\Sigma\,)$ consider only pure theories;
  i.e., without "`!`" and "`not(·)`"; rather than impure $\mathrm{DP}_{Imp}(\,\Sigma\,)$ and
- $\mathrm{DP}_{Prop}(\,\Sigma\,)$ deals with propositional logic, rather than predicate calculus, $\mathrm{DP}_{PC}(\,\Sigma\,)$.
  In this $\mathrm{DP}_{PC}(\,\Sigma\,)$ case, we seek only a *first* solution found; notice this corresponds to asking an impure query of the form "`foo(X, Y), !.`". (As propositional systems can only return at most one solution, this restriction is not meaningful in the propositional case.)

We will also combine subscripts, with the obvious meanings. When $\mathrm{DP}_\chi(\,\Sigma\,)$ is a special case of $\mathrm{DP}_\psi(\,\Sigma\,)$, finding that $\mathrm{DP}_\chi(\,\Sigma\,)$ is hard/non-approximatable immediately implies that $\mathrm{DP}_\psi(\,\Sigma\,)$ is hard/non-approximatable. Finally, each of the classes mentioned above allows an arbitrary number of modifications to the initial theory; *e.g.*, the set $\Sigma^{DR}(\mathrm{T})$ includes the theories formed by deleting *any* number of clauses, including the empty theory formed by deleting all of T's clauses. We let

- $\Sigma^{DR}_K(\mathrm{T})$ refer to the theories formed by deleting at most $K \in \mathcal{Z}^+$ clauses from T. We define $\Sigma^{AR}_K(\mathrm{T})$, $\Sigma^{OR}_K(\mathrm{T})$ and $\Sigma^{OA}_K(\mathrm{T})$ similarly.

3.1 Ordering of *Rules*

This subsection considers the challenge of re-ordering the rules, using the Σ^{OR} transformations. First, this task is intractable, even in trivial situations:

Theorem 2
Each of $\mathrm{DP}_{Perf,Imp,Prop}(\,\Sigma^{OR}\,)$ *and* $\mathrm{DP}_{Perf,Pur,PC}(\,\Sigma^{OR}\,)$ *is NP-complete.*

This theorem means that, unless $P = NP$, no poly-time algorithm can find a ordering of a list of impure proposition Horn clauses (resp., a list of pure predicate calculus Horn clauses) that returns the correct answers (resp., returns the correct *first* answer) to each of a given set of queries.

We can also restrict the space of possible theories by dealing only with theories formed by applying a limited number of "individual rule moves", where each such individual move will move a single rule to a new location. $\Sigma^{OR}_K(\mathrm{T})$ is then the set of theories formed by applying a sequence of at most K such individual moves. As a simple example, notice

$$\Sigma_1^{OR}(\{a,b,c,d\}) \quad = \quad \left\{ \begin{array}{lll} \{b,a,c,d\} & \{b,c,a,d\} & \{b,c,d,a\} \\ \{a,b,d,c\} & \{a,c,b,d\} & \{a,c,d,b\} \\ \{c,a,b,d\} & \{d,a,b,c\} & \{a,d,b,c\} \end{array} \right\}$$

includes only 9 of the $4! = 24$ possible permutations.

However, the task of finding the best re-ordering within this smaller space is also intractable:

Theorem 3
There is a $K \in \mathcal{Z}^+$ for which each of $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma_K^{OR})$ *and* $\mathrm{DP}_{Perf,Pur,PC}(\Sigma_K^{OR})$ *is NP-complete.*

This negative result shows the intractability of the obvious proposal of using a breath-first transversal of the space of all possible rule re-orderings, seeking the minimal set of changes that produces a perfect theory: First test the initial theory T_0 against the labeled queries, and return T_0 if it is 100% correct. If not, then consider all theories formed by applying only one single-move transformation, and return any perfect $\mathrm{T}_1 \in \Sigma_1^{OR}[\mathrm{T}_0]$. If there are none, next consider all theories in $\Sigma_2^{OR}[\mathrm{T}_0]$ (formed by applying pairs of moves), and return any perfect $\mathrm{T}_2 \in \Sigma_2^{OR}[\mathrm{T}_0]$; and so forth.

Approximatability: Many decision problems correspond immediately to optimization problems; for example, the IndependentSet decision problem (given a graph $G = \langle N, E\rangle$ and a positive integer K, is there a subset $M \subset N$ of at least $|M| \geq K$ nodes that are not connected to one another [GJ79, p194]) corresponds to the obvious maximization problem: Given a graph $G = \langle N, E\rangle$, find the largest independent subset of N. We can similarly identify the $\mathrm{DP}(\Sigma^{OR})$ decision problem with the "$\mathrm{MAX}(\Sigma^{OR})$" maximization problem: "Find the $\mathrm{T}' \in \Sigma^{OR}(\mathrm{T})$ whose accuracy is maximal".

Now consider any algorithm B that, given any $\mathrm{MAX}(\Sigma^{OR})$ instance $x = \langle \mathrm{T}, S\rangle$ with initial theory T and labeled training sample S, computes a syntactically legal, but not necessarily optimal, revision $B(x) \in \Sigma^{OR}(\mathrm{T})$. Then B's "performance ratio for the instance x" is defined as

$$MaxPerf(B,x) \;=\; MaxPerf_{\Sigma^{OR}}(B,x) \;=\; \frac{A(\,opt(x)\,)}{A(\,B(x)\,)}$$

where $opt(x) = opt_{Max(\Sigma^{OR})}(x)$ is the optimal solution for this instance; *i.e.*, $opt(x)$ is the theory $\mathrm{T}_{opt} \in \Sigma^{OR}(\mathrm{T})$ with maximal accuracy over S. (This $MaxPerf(B,x)$ value is arbitrarily large if $\mathrm{A}(\,B(x)\,) = 0$.)

We say a function $g(\cdot)$ "bounds B's performance ratio" iff

$$\forall\, x \in \mathrm{MAX}(\Sigma^{OR}), \quad MaxPerf(B,x) \;\leq\; g(|x|)$$

where $|x|$ is the size of the instance $x = \langle \mathrm{T}, S\rangle$, which we define to be the number of symbols in T plus the number of symbols used in S. Intuitively, this $g(\cdot)$ function indicates how closely the B algorithm comes to returning the best answer for x, over all $\mathrm{MAX}(\Sigma^{OR})$ instances x.

Now let $Poly(\,\mathrm{MAX}(\Sigma^{OR})\,)$ be the collection of all polytime algorithms that return legal answers to $\mathrm{MAX}(\Sigma^{OR})$ instances. It is natural to ask for the algorithm in $Poly(\,\mathrm{MAX}(\Sigma^{OR})\,)$ with the best performance ratio; this would indicate how close we can come to the optimal solution, using only a feasible computational time. For example, if this function was the constant 1 for $\mathrm{MAX}_{Imp,Prop}(\Sigma^{OR})$ then a poly-time algorithm could produce the optimal solution to any $\mathrm{MAX}(\Sigma^{OR})$ instance; as $\mathrm{DP}_{Imp,Prop}(\Sigma^{OR})$ is NP-complete, this would mean $P = NP$, which is why we do not expect to obtain this result. Or if this bound was some constant $c(x) = c \in \Re^+$, then we could efficiently obtain a solution within a factor of c of optimal, which may be good enough for some applications.[5]

However, not all problems can be approximated. Following [CP91, Kan92], we define

Definition 2 *A maximization problem* Max *is* NotPolyApprox *if there is a $\gamma \in \Re^+$ such that*

$$\forall B \in \mathrm{Poly}(\,\mathrm{Max}\,),\ \exists x \in \mathrm{Max},\quad MaxPerf_{\mathrm{MAX}}(\,B, x\,) \;\geq\; |x|^{\gamma}\ .$$

Arora et al. [ALM+92] prove that the "Maximum Independent Set maximization problem" is NotPolyApprox. We can use that result to prove:

Theorem 4
Unless $P = NP$, each of $\mathrm{MAX}_{Imp,Prop}(\Sigma^{OR})$ *and* $\mathrm{MAX}_{Pur,PC}(\Sigma^{OR})$ *is* NotPolyApprox.

As $|x|$ can get arbitrary large, this result means that these $\mathrm{MAX}(\Sigma^{OR})$ tasks cannot be approximated by any constant, nor even by any logarithmic factor nor any sufficiently small polynomial, etc.

3.2 Ordering of *Antecedents*

As mentioned above, each theory is an ordered list of rules, whose *antecedents are also ordered.* We can form new theories by re-ordering the antecedents of various rules, and note that these new theories can produce different answers to queries, in the impure contexts. We therefore let $\Sigma^{OA}(\mathrm{T})$ be the set of theories obtained by reordering the antecedents in T's rules, and ask the same questions asked above (sample complex-

[5]There are such constants for some other NP-hard optimization problems. For example, there is a polynomial-time algorithm that computes a solution whose cost is (essentially) within a factor of 11/9 for any MaxBinPacking maximization problem; see [GJ79, Theorem 6.2].

ity, computational complexity and approximatability). Here, we obtain the same results, *mutatis mutandis*:

First, note that $|\Sigma^{OA}(\mathrm{T})| = \prod_{c\in T}(\#\mathrm{Ants(c)})! = O(|\mathrm{T}|^{|T|})$, where $\#\mathrm{Ants(c)} \in \mathcal{Z}^{\geq 0}$ refers to the number of antecedents in the clause c. Re-using Theorem 1, this means we need only a polynomial number of samples.

Addressing the computational complexity of these tasks, we see

Theorem 5
Each of $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma^{OA})$, $\mathrm{DP}_{Perf,Pur,PC}(\Sigma^{OA})$, $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma^{OA}_K)$ *and* $\mathrm{DP}_{Perf,Pur,PC}(\Sigma^{OA}_K)$ *is NP-complete.*

(Notice this includes both the limited Σ^{OA}_K and unlimited Σ^{OA} transformations.)

Theorem 6
Unless $P = NP$, each of $\mathrm{MAX}_{Imp,Prop}(\Sigma^{OA})$ *and* $\mathrm{MAX}_{Pur,PC}(\Sigma^{OA})$ *is* NotPolyApprox.

3.3 Adding or Deleting Clauses

This subsection deals with adding or deleting clauses, in the impure contexts of finding all answers from impure programs, and finding the first answers from pure programs. We first state the results known about the standard pure context:

Theorem 7 (from [Gre95a])
In the pure context, for each $\Sigma \in \{\Sigma^{AR}, \Sigma^{DR}\}$,

- $\mathrm{DP}_{Perf,Prop,Pur}(\Sigma)$ *is NP-complete,*
- $\mathrm{MAX}_{Pur}(\Sigma)$ *is trivial to approximate:*
  $\exists B_\Sigma \in \mathrm{Poly}(\ \mathrm{MAX}_{Pur}(\Sigma)\ )$,
  $\forall x \in \mathrm{MAX}_{Pur}(\Sigma)$,
  $MaxPerf_{MAX_{Pur}(\Sigma)}(B, x) \leq 2.$

Hence, each of these maximization problems is trivially solved within a factor of 2 in the pure setting. However, in the impure setting:

Theorem 8
Unless $P = NP$, for each $\Sigma \in \{\Sigma^{AR}, \Sigma^{DR}\}$, each of $\mathrm{MAX}_{Imp,Prop}(\Sigma)$ *and* $\mathrm{MAX}_{Pur,PC}(\Sigma)$ *is* NotPolyApprox. *This holds even if the only clauses added (resp., deleted) are pure atomic literals.*

These same statements also apply to the tasks of adding or deleting antecedents: each can be trivially approximated in the pure context, but not in the impure contexts.

To address the sample complexity issue: Clearly $\ln(|\Sigma^{DR}|) = |\mathrm{T}|$ which means a polynomial number of samples is sufficient to make the familiar PAC-style guarantees. Similarly, $\ln(|\Sigma^{AR}|)$ is polynomial in the size of the theory and the language, in the propositional case. In the predicate calculus case, however, Σ^{AR} can be arbitrarily large. Observe however that $\Sigma^{AR}(\mathrm{T}) = \bigcup_k \Sigma^{AR}_k(\mathrm{T})$ and $\ln(|\Sigma^{AR}_k(\mathrm{T})|)$ is polynomial in $|\mathrm{T}|$. Hence, the number of samples required is polynomial in the "size" of the revised theory, which means we can apply the techniques from "nonuniform" pac-learning [BI88] to learn such classes.

4 Prioritizing Default Theories

This section first provides a brief motivation and introduction to prioritized Theorist-style default theories, then shows that our basic results apply here as well.

The conclusions reached by a "monotonic reasoning system" T will remain true even if additional information is added to the initial theory; *i.e.*, $\mathrm{T}(\sigma) \subseteq \mathrm{T}'(\sigma)$ whenever $\mathrm{T}' = \mathrm{T} \cup \{\rho\}$.[6] However, this monotonicity is often inappropriate; it often makes sense to allow a system T to produce the answers $\mathrm{T}(\sigma)$ to the query σ, but allow T′ to produce a completely different answer $\mathrm{T}'(\sigma) \neq \mathrm{T}(\sigma)$. As an example, knowing that something (`b`) is a bird, we may assume that `b` can fly. ...until we hear that `b` is a penguin; here we want to reverse that position and conclude that `b` can *not* fly. Unless, of course, we learn that `b` is in an airplane, etc etc etc. [Rei87]

The Theorist system [PGA87] provides one way to encode such information.[7] Its knowledge base consists of a set of facts $F = \{f_i\}$ which are the unchallengable claims and a set of "assumables" $D = \{d_j\}$, which correspond to the defaults. Here, perhaps,

```
       ⎧ bird(b51).
F1  =  ⎨ bird(b93). penguin(b93)
       ⎪ bird(X) :- penguin(X).
       ⎩ animal(X) :- bird(X).
       ⎧ assumable:  fly(B, yes) :- bird(B).
D1  =  ⎨ assumable:  fly(B, no) :- penguin(B).
       ⎩ assumable:  fly(B, yes) :- inPlane(B).
```

Given the query σ = `fly(b51, Q)`, Theorist will first try to prove this query using only its fact knowledge base F, using a pure Prolog-style (SLD) backward-chaining derivation. Notice this will fail. Theorist will next try to use the information in its collection of possible assumptions D_1: It is allowed to "assume" any relevant consistent clause "`assumable:`ρ" $\in D_1$: *i.e.*, Theorist first checks if ρ is consistent with F_1 and if so, it considers adding ρ in, forming $F' = F_1 \cup \{\rho\}$. Theorist next tries to prove σ using this new F'. Here, when Theorist finds the ρ = "`fly(B, yes) :- bird(B)`" clause, it will then add in ρ, and use it to return the answer `Yes`[Q/`yes`].

[6] Here, the "`No`" answer corresponds to the empty set {}.

[7] This description slightly modifies the Theorist syntax.

Now consider the query $\sigma' =$ `fly(b93, R)`. Here again THEORIST is unable to answer the query using only information in F_1, and so will consider its set of assumables, D_1. Unfortunately, there are now two assumables that qualify, which give contradictory answers to this question.

We can address this problem by assigning *priorities* to the defaults, with the understanding that lower priorities will be tried first [Bre89, vA90]. Here, the following priority assignment probably makes sense:

```
assumable 1:  fly(B, yes) :- inAirplane(B).
assumable 2:  fly(B, no)  :- penguin(B).
assumable 3:  fly(B, yes) :- bird(B).
```

which allows us to assume that anything `w` known to be in an airplane, is flying; otherwise (if we don't know that `w` is in an airplane), if `w` is a penguin, we can conclude `w` is not flying, and if not, and we know `w` is a bird, then we can conclude `w` does fly.

In general, a prioritized THEORIST-style default system can include many assumptions with the same priority number. Here, on the i^{th} iteration, the system will add in all of the priority-i assumptions. (in addition to assertions with priority-$i-1$, $i-2$, ..., 1). The initial collection of facts can therefore be regarded as priority-0.

Now suppose we have built a THEORIST knowledge base, with both facts and assumables, but have not assigned priorities to the assumables. (Or equivalently, we are not happy with the current set of priorities assigned.) We might then want to use a set of training samples to find the optimal priory assignment. Unfortunately,

Theorem 9
Unless $P = NP$, each of $\text{MAX}_{Imp,Prop}(\Sigma^{PD})$ *and* $\text{MAX}_{Pur,PC}(\Sigma^{PD})$ *is* NOTPOLYAPPROX. *This holds even if only two priority levels are considered.*

where $\Sigma^{PD}(\text{T})$ is the set of prioritized THEORIST-style theories that differs from the (prioritized THEORIST-style theory) T only by re-prioritizing the assumables. The subscripts χ in $\text{MAX}_\chi(\Sigma^{PD})$ describes properties of both knowledge bases (factual and assumable) of the underlying THEORIST theory.

5 Conclusion

5.1 Extensions

All of the previous theorems will hold even if we use a stochastic real-world oracle, encoded as $\mathcal{O}' : \mathcal{Q} \times \mathcal{A} \mapsto [0, 1]$, where the correct answer to the query q is a with probability $\mathcal{O}'(q, a)$. (Notice here that $a(\text{T}, q) = \mathcal{O}'(q, \text{T}(q))$.) Our deterministic oracle is a special case of this, where $\mathcal{O}'(q, a_q) = 1$ for a single $a_q \in \mathcal{A}$ and $\mathcal{O}'(q, a) = 0$ for all $a \neq a_q$.

There are obvious ways of extending our analysis to allow a more comprehensive accuracy function $a(\text{T}, \sigma)$ that could apply different rewards and penalties for different queries (*e.g.*, to permit different penalties for incorrectly identifying the location of a salt-shaker, versus the location of a stalking tiger [Vor91]). As these extensions lead to strictly more general situations, our underlying task (of identifying the optimal theory) remains as difficult; *e.g.*, it remains computationally intractable in general.

5.2 Contributions

Most theory revision systems deal with a particular set of theory-modification techniques (adding or deleting either a rule or an antecedent) that implicitly assumes the underlying theory is pure and the user is seeking all answers. Many reasoning contexts, however, violate these assumptions: theories are often impure, and many users seek only a subset of the answers. This paper presents two additional types of modifications that are meaningful for these "impure contexts" — *viz.*, re-ordering rules and re-ordering antecedents — and describes the complexities inherent in using them. In particular, it shows first that a polynomial number of training samples are sufficient to acquire the information needed to determine which transformation sequence is best. Unfortunately, however, the task of using this information to produce an optimal, or even near optimal, ordering of the rules (or ordering of the antecedents) is hopelessly intractable: no efficient algorithm can produce even a good approximation to the optimum. This resonants with earlier analyses of the theory revision task, and justifies the standard approach of hill-climbing to a locally-optimal theory. Wet also illustrate the additional complexities inherent in learning "impure" theories (beyond the problems of learning pure ones), by showing that the task of adding (resp., deleting) rules, which *is* trivially approximated in the pure context, is not approximatable in this setting. Finally, this paper also proves non-approximatable the task of determining the best priority values of a set of defaults.

A Proof Sketches of Some Theorems

Proof of Theorem 2 (sketch):[8] We reduce $\text{DP}_{Perf,Imp,Prop}(\Sigma^{OR})$ to 3SAT [GJ79, p259]: Determine if there is an variable assignment that satisfies a given 3-CNF formula (which is a conjunction of clauses, where each clause is a disjunction of at most three literals). By examining theories T_1 and T_2 (from Equations 1 and 2 respectively), notice that exactly one of `q` or `r` can be true as we reorder the

[8]In all proofs of the following "NP-complete"-claims, we prove only the the decision problem is NP-hard; each problem is trivially in NP.

clauses; we can therefore identify `q` with a positive literal, and `r` with a negative literal. Given any 3SAT formula $\varphi = \bigwedge_{j=1..m} c_j$ over the n literals $\{x_i\}_{i=1}^{n}$, we therefore form the theory T_φ which contains n copies of T_1's clauses (each using $\mathtt{q}_i$ and $\mathtt{r}_i$ for some $i = 1..n$) as well as the clause $\mathtt{c}_j$ `:- ` $\mathtt{r}_i$ whenever $x_i \in c_j$ and $\mathtt{c}_j$ `:- ` $\mathtt{q}_i$ whenever $\bar{x}_i \in c_j$. The query/answer set is

$$\{\langle \mathtt{c}_j;\ \mathtt{Yes}\rangle \quad \text{for } c_j \in \varphi\}.$$

Now observe there is a perfect re-ordering of T_φ's clauses iff φ has a satisfying assignment.

The proof for $\mathrm{DP}_{Perf,Pur,PC}(\Sigma^{OR})$ is a more cumbersome reduction to MONOTONE3SAT [GJ79, p259], which is an NP-complete special case of 3SAT where each clause contains either all positive, or all negative, literals. The theory $T_\varphi^{(PC)}$ includes both $\mathtt{u}_i$`(0)` and $\mathtt{u}_i$`(1)` for $i = 1..n$, as well as the clauses

```
cj(X) :-  uj1(V1), uj2(V2), uj3(V3),
          or3(V1, V2, V3, X).
```

for each clause $c_j \equiv x_{j1} \vee x_{j2} \vee x_{j3}$ containing only positive literals, and

```
cj(X) :-  uj1(V1), uj2(V2), uj3(V3),
          orN3(V1, V2, V3, X).
```

for each clause $c_j \equiv \bar{x}_{j1} \vee \bar{x}_{j2} \vee \bar{x}_{j3}$ containing only negative literals. $T_\varphi^{(PC)}$ also contains the 8 atomic clauses defining `or3` as `or3(A, B, C, D)` holds iff $D \equiv A \vee B \vee C$ (e.g., "`or3(1, 0, 1, 1)`" and "`or3(0, 0, 0, 0)`"), and the 8 atomic clauses defining `orN3` as `orN3(A, B, C, D)` iff $D \equiv \neg A \vee \neg B \vee \neg C$. The query/answer pairs are

$$S_\varphi^{(PC)} = \{\langle \mathtt{c}_j(X);\ \mathtt{Yes}[X/1]\rangle \mid \text{for } c_j \in \varphi\}\ .$$

Once again, there is a perfect ordering of $T_\varphi^{(PC)}$'s clauses iff φ is satisfiable. □

Proof of Theorem 3 (sketch): The proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma_K^{OR})$ is a reduction to X3C (EXACT COVER BY 3-SETS [GJ79, p221]): Given a set of elements $X = \{x_i\}_{i=1}^{3K}$ and collection of 3-element subsets $C = \{c_j\}_{j=1}^{M}$ of elements of X, determine if there is a subcollection $C' \subset C$ of disjoint sets whose union covers all of X. Given any such X and C, let

$$T_{XC} = \left\{ \begin{array}{ll} \mathtt{x}_i\ \mathtt{:-}\ \mathtt{c}_j. & \text{when } x_i \in c_j \\ \left.\begin{array}{l} \mathtt{c}_j\ \mathtt{:-}\ \mathtt{!,\ fail.} \\ \mathtt{c}_j. \end{array}\right\} & \text{for } c_j \in C \end{array} \right\}$$
$$S_{XC} = \{\langle \mathtt{x}_i;\ \mathtt{Yes}\rangle \quad \text{for } i = 1..3K\}\ .$$

By moving any "$\mathtt{c}_j$`.`" atomic clause forward, we can cause $\mathtt{c}_j$ to become entailed, which in turn means the resulting theory will entail the associated $\mathtt{x}_i$'s. There is an exact cover of $\langle X, C\rangle$ iff we can form a perfect theory by re-ordering exactly K of the "$\mathtt{c}_j$`.`" clauses.

The proof for $\mathrm{DP}_{Perf,Pur,PC}(\Sigma_K^{OR})$ is also a reduction to X3C, and re-uses Theorem 2's trick of explicitly encoding a disjunction. Here, the initial theory $T_{XC}^{(PC)}$ includes, for each set c_j, the atomic clauses $\mathtt{c}_j$`(0)` and $\mathtt{c}_j$`(1)`, in this order; and for each x_i, a clause of the form

```
xi(Zk) :-  ci1(Y1), ci2(Y2), or2(Y1, Y2, Z2),
           ci3(Y3), or2(Z2, Y3, Z3), ...,
           cik(Yk), or2(Zk-1, Yk, Zk).
```

where the antecedent $\mathtt{c}_{j\ell}(\cdot)$ appears whenever $x_i \in c_{j\ell}$. (Here, x_i is in exactly k sets; of course, different x_i's will be in differing numbers of sets.) $T_{XC}^{(PC)}$ also includes the four clauses that define `or2(A, B, C)` as $C \equiv A \vee B$. The query/answer set is

$$S_{XC}^{(PC)} = \{\langle \mathtt{x}_i(W);\ \mathtt{Yes}[W/1]\rangle \mid \text{for } x_i \in X\}\ .$$

Now observe the underlying X3C problem has a solution iff there is a set of K reorderings that produces a perfect theory. □

Proof of Theorem 5 (sketch): The proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma^{OA})$ is essentially the same as the proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma^{OR})$ (Theorem 2), using the observation that reordering the antecedents of "`q :- !, fail.`" to form "`q :- fail, !.`" has the effect of allowing `q` to be entailed. The proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma_K^{OA})$ is similarly related to the proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Sigma_K^{OR})$ (Theorem 3), as changing "$\mathtt{c}_j$ `:- !, fail.`" to "$\mathtt{c}_j$ `:- fail, !.`" causes $\mathtt{c}_j$ to be entailed.

For $\mathrm{DP}_{Perf,Pur,PC}(\Sigma^{OA})$, replace each of (Theorem 2) $T_\varphi^{(PC)}$'s "$\mathtt{u}_j$`(0).`" and "$\mathtt{u}_j$`(1).`" pair of clauses with the single clause "$\mathtt{u}_j$`(Y) :- prefer0(Y), prefer1(Y).`", and also include the four atomic clauses

$$\begin{array}{ll} \mathtt{prefer0(0).} & \mathtt{prefer0(1).} \\ \mathtt{prefer1(1).} & \mathtt{prefer1(0).} \end{array} \qquad (4)$$

in this (left-to-right) order. Notice the first answer returned to the (sub)query "$\mathtt{u}_j$`(Y)`" is $\mathtt{Yes}[Y/0]$, when using the initial "$\mathtt{u}_j$`(Y) :- prefer0(Y), prefer1(Y).`" clause, but if we re-order the clause's antecedents to "$\mathtt{u}_j$`(Y) :- prefer1(Y), prefer0(Y).`", we get $\mathtt{Yes}[Y/1]$ for this subquery. The rest of the proof is identical to the proof that $\mathrm{DP}_{Perf,Pur,PC}(\Sigma^{OR})$ is NP-hard in Theorem 2.

The proof for $\mathrm{DP}_{Perf,Pur,PC}(\Sigma_K^{OA})$ follows from the proof of Theorem 3, using this same trick of by replacing each pair $\{\mathtt{c}_j\mathtt{(0).},\ \mathtt{c}_j\mathtt{(1).}\}$ with the single clause "$\mathtt{c}_j$`(Y) :- prefer0(Y), prefer1(Y).`" and by including the four atomic clauses in Equation 4. As above, we can reorder the "`prefer0`" and "`prefer1`" literals of the "$\mathtt{c}_j$`(Y) :- prefer0(Y), prefer1(Y).`" clauses to get different answers to the "$\mathtt{c}_j$`(Y)`" subquery; etc. □

References

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy.

Proof verification and hardness of approximation problems. In *FOCS*, 1992.

[BCH90] E. Boros, Y. Crama, and P.L. Hammer. Polynomial-time inference of all valid implications for horn and related formulae. *Annals of Mathematics and Artificial Intelligence*, 1:21–32, 1990.

[BGT93] Francesco Bergadeno, Daniele Gunetti, and Umberto Trinchero. The difficulties of learning logic programs with cut. *Journal of AI Research*, 1:91–107, 1993.

[BI88] G. Benedek and A. Itai. Nonuniform learnability. In *ICALP-88*, pages 82–92, 1988.

[Bre89] Gerhard Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *IJCAI-89*, pages 1043–48, Detroit, August 1989.

[CM81] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog*. Springer-Verlag, New York, 1981.

[CP91] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, 93(2):241–62, 1991.

[DP91] Jon Doyle and Ramesh Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48(3), 1991.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.

[Gre91] Russell Greiner. Finding the optimal derivation strategy in a redundant knowledge base. *Artificial Intelligence*, 50(1):95–116, 1991.

[Gre95a] Russell Greiner. The challenge of revising impure theories. Technical report, Siemens Corporate Research, 1995. (See also `ftp://scr.siemens.com/pub/learning/Papers/greiner/impure.ps`).

[Gre95b] Russell Greiner. The complexity of theory revision. In *IJCAI-95*, 1995. (See also `ftp://scr.siemens.com/pub/learning/Papers/greiner/comp-tr.ps`).

[Gro91] Benjamin Grosof. Generalizing prioritization. In *KR-91*, pages 289–300, Boston, April 1991.

[Kan92] Viggo Kann. *On the Approximability of NP-Complete Optimization Problems*. PhD thesis, Royal Institute of Technology, Stockholm, 1992.

[LDRG94] Pat Langley, George Drastal, R. Bharat Rao, and Russell Greiner. Theory revision in fault hierarchies. In *Fifth International Workshop on Principles of Diagnosis (DX-94)*, New Paltz, NY, 1994.

[Lev84] Hector J. Levesque. Foundations of a functional approach to knowledge representation. *Artificial Intelligence*, 23:155–212, 1984.

[LV91] Charles Ling and Marco Valtorta. Revision of reduced theories. In *MLC-91*, pages 519–23, 1991.

[MB88] S. Muggleton and W. Buntine. Machine invention of first order predicates by inverting resolution. In *MLC-88*, pages 339–51. Morgan Kaufmann, 1988.

[OM94] Dirk Ourston and Raymond J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66(2):273–310, 1994.

[PGA87] David Poole, Randy Goebel, and Romas Aleliunas. Theorist: A logical reasoning system for default and diagnosis. *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–52, New York, 1987. Springer Verlag.

[Rei87] Raymond Reiter. Nonmonotonic reasoning. In *Annual Review of Computing Sciences*, volume 2, pages 147–87. Annual Reviews Incorporated, Palo Alto, 1987.

[vA90] Paul van Arragon. Nested default reasoning with priority levels. In *CSCSI-90*, pages 77–83, Ottawa, May 1990.

[Val89] Marco Valtorta. Some results on the complexity of knowledge-base refinement. In *MLC-89* pages 326–31, 1989.

[Val90] Marco Valtorta. More results on the complexity of knowledge base refinement: Belief networks. In *MLC-90*, pages 419–26, 1990.

[Vap82] V.N. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, New York, 1982.

[Vor91] David Vormittag. Evaluating answers to questions, May 1991. Bachelors Thesis, University of Toronto.

[WM94] David C. Wilkins and Yong Ma. The refinement of probabilistic rule sets: sociopathic interactions. *Artificial Intelligence*, 70:1–32, 1994.

[WP93] James Wogulis and Michael J. Pazzani. A methodology for evaluating theory revision systems: Results with Audrey II. In *IJCAI-93*, pages 1128–1134, 1993.