The Complexity of Revising Logic Programs*

Russell Greiner†
Department of Computing Science
University of Alberta
Edmonton, Alberta T6G 2H1
greiner@cs.ualberta.ca

January 25, 1999

Abstract

A rule-based program will return a set of answers to each query. An *impure* program, which includes the Prolog `cut` "!" and "`not(·)`" operators, can return different answers if its 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 paired 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 distributional 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 next 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 *new rules to add*; and (3) the optimal set of *existing rules to delete.*

Keywords: Theory Revision, Inductive Logic Programming,
Computational Complexity & Approximatability, PAC-Learning

---

*This paper extends the short article "The Challenge of Revising an Impure Theory" that appears in the *Proceedings of the Twelfth International Conference on Machine Learning*, Lake Tahoe, July 1995.

†I gratefully acknowledge the many helpful comments I received from the anonymous reviewers. Much of this work was done while I worked at Siemens Corporate Research, in Princeton, NJ.

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 (also known as its "theory") contains incorrect or mis-organized information. In some situations, we will be able to obtain the correct answers to the queries — *e.g.*, these answers may be supplied by an human expert who was called when expert system returned an answer that was found to be incorrect (*e.g.*, if the proposed repair does not correct a device's fault), or perhaps these answers are known by the programmer, debugging his code (see Subsection 1.1 below). Here, we would like to use these query/correct-answer pairs to produce a theory that is (more nearly) correct.

A typical "Inductive Logic Programming" (ILP) system would use *only* this set of correctly-answered queries to produce a new, more accurate theory. If the initial theory T_0 was already very accurate (which is typically the case when T_0 is part of a deployed system), the ILP algorithm would in effect have to re-learn most of T_0; this seems very wasteful. Instead, it is often more efficient to correct T_0. *Theory revision* is the process of using these correctly-answered queries to *modify* the given initial theory, to produce a new, more accurate theory.

Many implemented theory revision systems hill-climb 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. An alternative class of transformations *re-arrange* the order of the rules, or of the antecedents. These transformations can 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, prioritized default theories [6, 29], most production systems [33, 20], as well as PROLOG [8].

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. Section 3 then 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 4 then presents our main results, showing first that this task is tractable if the initial theory is "syntactically close" to the optimal theory, but then that this task becomes intractable[2] in other trivial situations — *e.g.*, even if each query is 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 and k-Horn. This also demonstrates the intractability of finding the smallest number of "individual re-orderings" required to produce a perfect ordering.

We next deal with the "agnostic" version of this task [32]: asking for the *most accurate*

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

[2] Throughout, we will assume that $P \neq NP$ [24], which implies that any NP-hard problem is intractable. This also implies certain approximation claims, presented below. Also, we will define below the terms used in this section, including "syntactically close" and "k-Horn".

reordering, in cases where perhaps no reordering will produce a perfect theory. We prove that the agnostic task is not even approximable; *i.e.*, unless $P = NP$, no polynomial-time algorithm can identify an ordering of an n-rule theory whose accuracy is within n^γ of optimal, for some $\gamma > 0$. (As this result applies to arbitrarily large theories, this means no polynomial-time algorithm can identify an ordering that is within any constant, or any logarithmic function, of optimal.) 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. The appendix provides complete proofs of the theorems, to augment the sketches that appear within the main text.

We first close this introduction by first mentioning two other obvious applications of this framework, and then describing related research, including the work in "Inductive Logic Programming" and "belief revision".

1.1 Other Uses of Theory Revision

Anytime ILP: As mentioned above, typical inductive logic programs build a logic program *from scratch*, based only on a set of training examples that exhibit the desired behavior of the program. Most such programs assume access to a sufficient number of correct training examples to determine the appropriate logic program.

In some situations, however, one may need to produce and use a program before obtaining such resources. Here, one may want an "anytime" algorithm [4] that can, at any time, return an adequate program. (Of course, later programs, based on more samples, will usually be superior.) A naïve implementation for such a system would start from scratch each time a program is requested; Given m samples, it would run an ILP system to produce the program T_m; and later, when given k more samples, it would run this ILP system on (only) the $m+k$ samples to produce the program T_{m+k}. This is clearly wasteful, as the algorithm would be forced to re-learn the "correct parts" of the program each time. A better approach would use the additional k samples to improve the stored T_m program.

Of course, this requires an algorithm that can take an initial program, together with a set of samples, and produce a superior program; notice theory revision systems are designed to do exactly this task.

Debugging Logic Programs: While we earlier worded our revision task as improving a deployed knowledge-based system, another obvious application is debugging code in general: Few people are able to directly write perfect code; instead, most write code that seems about right, and then "try it out" on some test cases, whose behavior they wish to match. That is exactly the task being considered here.

Our results specify how many test cases should be used, for each of the classes of modifications being considered; they also show that this task is (trivially) feasible if the current program has only a few bugs. We then prove the underlying task is extremely difficult if the original program is very buggy, by proving that no theory reviser (be it a computer program, or a human programmer) can efficiently find even a near-optimal revision in such situations. (Indeed, here it may seem better to simply throw out the original program and start afresh; but see the negative results from Inductive Logic Programming [10, 11].)

As specific evidence that people who write *logic* programs often use such debugging techniques, please note that this is an essential step in building rule-based systems, where it has been shown to work effectively; *cf.*, texts on Knowledge Acquisition [41].

1.2 Related Research

A theory revision process "*learns from examples*", as it uses "labeled samples" (here, correctly-labeled queries) to produce an accurate theory [15]. As the resulting "concept" is a logic program, such processes fits within the sub-topic of "*Inductive Logic Programming* (ILP)" [39]. Most ILP systems, however, consider only adding new information to an initial (often empty) starting theory; by contrast, theory revision systems consider other ways of modifying an existing, not-necessarily-empty initial theory, often including rule- or antecedent-deletion.

There are many *implemented theory revision systems*, including AUDREY [44], FONTE [38], EITHER [40] and DELTA [34]. Most of these system 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 DELTA system is an exception, as it does reorder the rules. The empirical results discussed in [34] show that such transformations can be used effectively.

There are a variety of related *complexity results*. (1) The companion paper [28, 27] 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 the pure context. (2) Valtorta and Ling [36, 37] 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. (3) Wilkins and Ma [43] 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 propositional case, when all rules have unit weight and a single successful rule is sufficient to establish a conclusion. (4) There are a number of results on the complexity of (PAC-)learning logic programs *from scratch* (*i.e.*, of the ILP task); *cf.*, [10, 9, 11, 18]. We outlined above how our framework is different. Note also that we focus on Horn theories that are syntactically close to an initial theory; by contrast, most ILP systems can return any Horn theory. (Although by construction, they tend to return theories which are syntactically close to the empty theory — *i.e.*, small programs.)

Bergadano *et al.* [3] also considers the challenges of *learning* impure *logic programs* (which can include the PROLOG cut "`!`" and "`not(·)`" operators), noting that it can be more difficult than learning pure programs. Our paper gives additional teeth to this claim, by showing specific tasks (*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 are not approximatable for impure programs — see Theorems 8 and 9 below.

This paper has some superficial similarities with [26], as both articles consider the complexity of (in essence) finding the best ordering of a set of rules. However, while [26] deals with the *efficiency* of finding *any* answer to a given query, this paper deals with the *accuracy* of the particular answer returned.

In some situations, there may be no rearrangement of the clauses that is "perfect"; *i.e.*, which entails all the positively-labeled queries, and none of the negatively-labeled queries. Here, we seek the "optimal arrangement" (*i.e.*, with the highest accuracy); this corresponds exactly to the "*agnostic learning*" model. Kearns, Schapire and Sellie [32] also show that a particular agnostic learning task is intractable. Our results differ by dealing with a different class of "samples" (arbitrary queries, not bit vectors), and by having a different class of hypotheses (predicate calculus Horn theories, rather than propositional conjunctions). More significantly, we present situations where the computational task is not just intractable, but is not even approximatable.

Like theory revision systems, *belief revision* systems [1, 13, 23, 31] also modify a given theory to incorporate some new observations about the world. Such formalisms take as input an initial theory T_0 and a new assertion $\langle q, +\rangle$, (resp., new retraction $\langle r, -\rangle$) and return a new (consistent) theory T' that entails q (resp., does not entail r) but otherwise is "close" to T_0 [13]. Most belief revision frameworks provide an axiomatic description of the preferred revision, which explicitly prefers a theory that is *"semantically close"* to the initial theory, and which does/does-not entail a *single* new proposition [13]. In general, the resulting revised theory will not depend on the syntactic structure of the initial theory — *i.e.*, if $\mathrm{T}_1 \equiv \mathrm{T}_2$, then the theory obtained by revising T_1 with the assertion $\langle q, +\rangle$ is equivalent to the theory obtained by revising T_2 with $\langle q, +\rangle$.

Belief revision systems typically use only a *single labeled query* to modify an initial theory T_0, seeking a theory *close to* T_0 which correctly does/does-not entail that query.[3] By contrast, theory revision uses a *set of labeled queries* when modifying T_0, searching within the space of theories that are *syntactically close* to T_0 for a theory with *optimal accuracy* with respect to those queries. Notice a theory revision system (1) does not require that the revised theory be correct for any specific labeled query, and (2) may produce different theories from semantically equivalent initial theories (as it may search different spaces of theories). As a final distinction, we show that the theory revision task is difficult even if both initial and final theories (as well as the queries) are propositional and k-Horn; by contrast, many belief revision frameworks deal with arbitrary predicate-calculus formulae. (Of course, the standard belief revision tasks — *e.g.*, the "counterfactual problem" — are complete for higher levels in polynomial-time hierarchy [19].)

[3]While the work on "iterated revision" [5, 25, 21, 14] also considers more than a single assertion, it usually deals with a *sequence* of assertions, where each new assertion must be incorporated, as it arrives. Afterwards, it is no longer distinguished from any other information in the present theory (but see [22]). We, however, consider the assertions as a *set*, which is seen at once, and whose elements need not all be incorporated.

2 Framework

Section 2.1 first describes our task within the context of propositional PROLOG programs. Section 2.2 then extends this description to predicate calculus, and Section 2.3 presents several further generalizations of our framework.

2.1 Propositional Horn Theories

We define a "theory" as an ordered list of Horn clauses (also known as "rules"), where each clause includes at most one positive literal (the "head") and an ordered list of zero or more literal antecedents (the "body"), all over a finite language. Such a theory is "k-Horn" if each of its clauses contain at most k literals. A theory is "impure" if it includes any rule whose antecedents use either the PROLOG cut "`!`" or negation-as-failure "`not(·)`" operator. See Clocksin&Mellish [8] 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 a theory's rules, and each rule's antecedents, in a particular order; and on reaching a cut antecedent within a rule, PROLOG will not consider any of the other rules whose heads unify with the current subgoal.

As a trivial example, consider the theory

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

Given the query "`q`", PROLOG first finds the rules whose respective heads unify with this goal (which are the first two rules in Equation 1), and processes them in the top-to-bottom order shown. 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.*,

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

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

Borrowing from [35, 17], we also view a theory T as a function that maps each query to its proposed answer; hence, T: $\mathcal{Q} \mapsto \mathcal{A}$, where $\mathcal{Q}$ is a (possibly infinite) set of queries, and $\mathcal{A} = \{$ `Yes`, `No` $\}$ 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 with respect to the oracle $\mathcal{O}$):

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

Hence, $a(\mathrm{T}_1,\,\text{“q”}) = 1$ as T_1 provides the correct answer $\mathcal{O}(\,\mathtt{q}\,) = \mathtt{No}$, 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 $Pr\colon \mathcal{Q} \mapsto [0,1]$, where $Pr(\sigma)$ is the probability that the query σ will be posed.[4] 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, $\Upsilon(\mathrm{T}) = \{\mathrm{T}_i\}$, where each such $\Upsilon(\mathrm{T})$ contains the set of theories formed by applying various transformations to a given theory T; for example, $\Upsilon^{Ord-Rules}(\mathrm{T})$ 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 $\mathrm{T}_{opt} \in \Upsilon(T)$ whose expected accuracy is maximal;[5] *i.e.*,

$$\forall\, \mathrm{T}' \in \Upsilon(\mathrm{T})\colon\ \mathrm{A}(\,\mathrm{T}_{opt}\,) \;\geq\; \mathrm{A}(\,\mathrm{T}'\,)\ . \tag{3}$$

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 3 below discusses the number of samples required to obtain the information needed to identify a good $\mathrm{T}^* \in \Upsilon(\mathrm{T})$, with high probability.

We are then left with the challenge of computing the best theory, once given these samples. Section 4 addresses the computational complexity of this process, showing that the task is not just intractable, 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.

---

[4] A distribution is "stationary" if it does not change over time; here this means that $Pr(\cdot)$ is a function.

[5] While "maximal accuracy" is equivalent to "minimal error", these two descriptions lead to different approximatability results. We word our claims in terms of "accuracy" to be compatible with our approximatability results.

2.2 Predicate Calculus

To handle predicate calculus expressions, we must consider answers of the form $\texttt{Yes}[\{X_i = \texttt{v}_i\}]$, where the expression within the brackets is a binding list of the free variables, corresponding to the *first* answer found to the query.[6] For example, given the theory

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

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

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

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

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

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

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

(here the system had to backtrack).

As a second example, we will later use the theory:

$$\mathrm{T}_{ab} \quad = \quad \left\{ \begin{array}{llll} \multicolumn{4}{l}{\texttt{aORb(Z) :- a(X), b(Y), or2(X, Y, Z).}} \\ \texttt{a(0).} & \texttt{a(1).} & \texttt{b(0).} & \texttt{b(1).} \\ \texttt{or2(0, 0, 0).} & \texttt{or2(0, 1, 1).} & \texttt{or2(1, 0, 1).} & \texttt{or2(1, 1, 1).} \end{array} \right\} \tag{4}$$

Here the query `aORb(Z)` will return the answer

$$\mathrm{T}_{ab}(\texttt{aORb(Z)}) \quad = \quad \texttt{Yes}[Z = \texttt{0}]$$

as `a(0)` comes before `a(1)`, and `b(0)` comes before `b(1)`. Notice a theory that inverts the order of either of these would instead return, as its first answer, $\texttt{Yes}[Z = \texttt{1}]$.

2.3 Extensions

All of the theorems in this paper 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 $a(\mathrm{T}, q) = \mathcal{O}'(q, \mathrm{T}(q))$.) 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 $a \neq a_q$.

[6]Following PROLOG's conventions, we will capitalize each variable, as in the "X_i" above. Also, to simplify our notation, if only a single variable is bound, we will omit the $\{\ldots\}$ braces and simply write $\texttt{Yes}[X = \texttt{v}]$. Moreover, if there are no variables involved, we will write simply `Yes`.

There are obvious ways of extending our analysis to allow a more comprehensive accuracy function $a(\mathrm{T}, \sigma)$ 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). We also contrast the task of finding the *first* answer with finding *all* answers; clearly we can also consider the task of finding the first *two* answers, or in general, of seeking the first k answers to a query. 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, and non-approximatable, in general.

3 Sample Complexity

This short section considers how many training samples are required to obtain the information needed to identify a good $\mathrm{T}^* \in \Upsilon(\mathrm{T})$ with high probability, as a function of the space of theories $\Upsilon(\mathrm{T})$ being considered.

As mentioned above, a "training sample" $S = \{\langle \sigma_i, \mathcal{O}(\sigma_i) \rangle\}$ is a (finite) multiset of specific "labeled queries", each of which is a query paired with its correct answer. Given such a training sample, we define the "empirical accuracy" of a theory T, written $\bar{\mathrm{A}}_S(\mathrm{T})$, as

$$\bar{\mathrm{A}}_S(\mathrm{T}) \quad = \quad \frac{1}{|S|} \sum_{\sigma_i \in S} a(\mathrm{T}, \sigma_i)$$

Notice $\bar{\mathrm{A}}_S(\mathrm{T}) \in [0,1]$; moreover, the Law of Large Numbers guarantees that this quantity will approach T's true accuracy $\mathrm{A}(\mathrm{T})$ as the sample size grows large (with probability 1). Many standard statistical tools bound the probability that $\bar{\mathrm{A}}_S(\mathrm{T})$ will be far from $\mathrm{A}(\mathrm{T})$, as a function of sample size. We can use such a tool to derive [7]:

Theorem 1 (from [42, Theorem 6.2])
Given a class of theories $\Upsilon = \Upsilon(\mathrm{T})$ *and constants* $\epsilon, \delta > 0$*, let* $\mathrm{T}_* \in \Upsilon$ *be the theory with the largest empirical accuracy after*

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

samples (each a labeled query), drawn from the stationary distribution, $Pr(\cdot)$*. Then, with probability at least* $1-\delta$*, the expected accuracy of* T_* *will be within* ϵ *of the optimal theory in* Υ*; 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 a $1-\epsilon$-good theory from Υ with probability at least $1-\delta$, whenever $\ln(|\Upsilon|)$ is polynomial in the relevant parameters. Notice this is true for $\Upsilon = \Upsilon^{Ord-Rules}(\mathrm{T})$: Using Stirling's Formula, $\ln(|\Upsilon^{Ord-Rules}(\mathrm{T})|) = O(n \ln(n))$, which is polynomial in the size of the initial theory $n = |\mathrm{T}|$. We will see that (a variant of) this "$\ln(|\Upsilon|) = \mathrm{poly}(|\mathrm{T}|)$" claim is true for essentially every class of theories Υ considered in this paper.

T	= a theory; *i.e.*, a set of (possibly impure) Horn clauses
Functions Υ^{χ} mapping a theory T *to set of theories* $\Upsilon^{\chi}(\mathrm{T})$	
$\Upsilon^{Ord-Rules}(\mathrm{T})$	= set of theories formed by re-ordering clauses of theory T
$\Upsilon^{Ord-Antes}(\mathrm{T})$	= set of theories formed by re-ordering antecedents of T's clauses
$\Upsilon^{Add-Rules}(\mathrm{T})$	= set of theories formed by adding new clauses to T
$\Upsilon^{Del-Rules}(\mathrm{T})$	= set of theories formed by deleting existing clauses from T
For any Υ^{χ} that maps a theory to a set of theories:	
$\Upsilon^{\chi}_{K}(\mathrm{T})$	= set of theories formed by applying sequences of at-most-K χ-modifications
	Note $K = K(\lvert \mathrm{T} \rvert)$ may be a function of the size of the initial theory T
Decision Problem, for any $\Upsilon = \Upsilon^{\chi}$ that maps a theory to a set of theories:	
$\mathrm{DP}(\Upsilon)$	= Decision problem defined in Definition 1
$\mathrm{DP}_{Perf}(\Upsilon)$	= $\mathrm{DP}(\Upsilon)$ with $p = 1$
Gen'l: $\mathrm{DP}_{Opt}(\Upsilon)$	allows arbitrary p
$\mathrm{DP}_{Pur}(\Upsilon)$	= $\mathrm{DP}(\Upsilon)$ with pure theories
Gen'l: $\mathrm{DP}_{Imp}(\Upsilon)$	allows impure theories
$\mathrm{DP}_{Prop}(\Upsilon)$	= $\mathrm{DP}(\Upsilon)$ with propositional theories
Gen'l: $\mathrm{DP}_{PC1}(\Upsilon)$	allows predicate calculus, seeking only the *first* answer
Gen'l: $\mathrm{DP}_{PC-All}(\Upsilon)$	allows predicate calculus, seeking *all* answers
Optimization Problem, for any $\Upsilon = \Upsilon^{\chi}$ that maps a theory to a set of theories:	
$\mathrm{MAX}_{\rho}(\Upsilon^{\chi})$	= maximization problem, with "constraints" $\rho \subset$ {Perf, Pur, Prop, ...} (see above)

Table 1: Definitions and Notation

4 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 true 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:

> $\Upsilon^{Ord-Rules}(\mathrm{T})$ (resp., $\Upsilon^{Ord-Antes}(\mathrm{T})$, $\Upsilon^{Add-Rules}(\mathrm{T})$ and $\Upsilon^{Del-Rules}(\mathrm{T})$) 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 $\Upsilon \in \{\, \Upsilon^{Ord-Rules},\ \Upsilon^{Ord-Antes},\ \Upsilon^{Add-Rules},\ \Upsilon^{Del-Rules} \,\}$ is a function mapping a theory to a set of theories. These terms, as well as our other notation, is summarized in Table 1.

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

Definition 1 ($\mathrm{DP}(\Upsilon)$ Decision Problem)

INSTANCE:
- *Initial theory* T*;*
- *Labeled training sample* $S = \{\langle q_i, \mathcal{O}(q_i)\rangle\}$ *containing a set of labeled queries; and*
- *Accuracy value* $p \in [0,1]$.

QUESTION: *Is there a theory* $\mathrm{T}' \in \Upsilon(\mathrm{T})$ *such that*

$$A(\mathrm{T}') = \frac{1}{|S|} \sum_{\langle q_i, \mathcal{O}(q_i)\rangle \in S} a(\mathrm{T}', q_i) \quad \geq \quad p\ ?$$

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

We will also consider the following special cases:

- $\mathrm{DP}_{Perf}(\Upsilon)$ requires that $p = 1$, *i.e.*, seeking perfect theories; rather than "optimal" theories $\mathrm{DP}_{Opt}(\Upsilon)$;
- $\mathrm{DP}_{Pur}(\Upsilon)$ consider only pure theories, *i.e.*, without "`!`" and "`not(·)`"; rather than impure $\mathrm{DP}_{Imp}(\Upsilon)$ and
- $\mathrm{DP}_{Prop}(\Upsilon)$ deals with propositional logic, rather than predicate calculus, $\mathrm{DP}_{PC1}(\Upsilon)$. The "1" in the "PC1" subscript is used to emphasize the fact that we are only seeking the *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 later consider $\mathrm{DP}_{PC-All}(\Upsilon)$, which seeks *all* answers to each query.

We will combine subscripts, with the obvious meanings; hence in general we will write $\mathrm{DP}_{A,B,C}(\Upsilon^\dagger)$ where $A \in \{\text{Perf, Opt}\}$, $B \in \{\text{Pur, Imp}\}$, and $C \in \{\text{Prop, PC1, PC-All}\}$. Most of our results deal with either the $\{A, \text{Imp, Prop}\}$, or the $\{A, \text{Pur, PC1}\}$, context.

When $\mathrm{DP}_\chi(\Upsilon)$ is a special case of $\mathrm{DP}_\psi(\Upsilon)$, finding that $\mathrm{DP}_\chi(\Upsilon)$ is hard/non-approximatable immediately implies that $\mathrm{DP}_\psi(\Upsilon)$ 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 $\Upsilon^{Del-Rules}(\mathrm{T})$ includes the theories formed by deleting *any* number of clauses, including the empty theory formed by deleting all of T's clauses. We let

> $\Upsilon_K^{Del-Rules}(\mathrm{T})$ refer to the theories formed by deleting at most $K \in \mathcal{Z}^+$ clauses from T. We similarly define $\Upsilon_K^{Add-Rules}(\mathrm{T})$ (resp., $\Upsilon_K^{Ord-Rules}(\mathrm{T})$ and $\Upsilon_K^{Ord-Antes}(\mathrm{T})$) as the set of theories formed by adding at most K new clauses (resp., moving at most K clauses to new positions, and moving each of at most K antecedents to a new position in the same clause). In a slight abuse of notation, we can let K be a function $K(|\mathrm{T}|)$ of the size of the initial theory T.

N.b., all of our negative results hold for k-Horn theories, where k is a small constant (in each case, bounded by 6). Moreover, we only consider "consistent training samples": that is, in each case, there is a k-Horn theory that can correctly label all of the training queries. That theory, however, is not always within the space of theories being considered. Third, as our $\Upsilon^{Add-Rules}(\mathrm{T})$ and $\Upsilon_K^{Add-Rules}(\mathrm{T})$ tasks each involve adding new rules, they clearly resemble the more typical "Inductive Logic Programming" task, which is known to be hard [10, 11]. Our results, however, apply even if we consider only adding in *atomic* literals, rather than more general clauses. Finally, note that computing each $a(\mathrm{T}', q_i)$ implicitly requires computing $\mathrm{T}'(q_i)$, which can be expensive for expressive theories. However, in the results that follow, we will assume that there is an efficient way to compute $a(\mathrm{T}', q_i)$. This is always true when T' is a propositional Horn theory and q_i is atomic [16], which is our main focus. Otherwise, we can assume another oracle that in constant time returns this $a(\mathrm{T}', q_i)$ value.

4.1 Ordering of *Rules*

This subsection considers the challenge of re-ordering the rules, using the $\Upsilon^{Ord-Rules}$ transformations. First, this task is intractable even in trivial situations:

Theorem 2
Each of $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Rules})$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon^{Ord-Rules})$ *is NP-complete.*

Proof (sketch): The main insight required for the $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Rules})$ proof is suggested by the T_1 and T_2 theories, shown in Equations 1 and 2: As exactly one of `q` or `r` holds in each theory, we can view `r` as not-`q` (*i.e.*, $\mathtt{r} \equiv \overline{\mathtt{q}}$). Moreover, the assignment to this "literal" (*i.e.*, whether `q` or $\mathtt{r} \equiv \overline{\mathtt{q}}$ holds) depends on the order the two `q`-headed clauses. We can now show NP-hardness by reducing an arbitrary 3SAT problem with n literals and m clauses to a theory formed with n such "mini-theories" (each with a copy of the three rules shown in Equation 1, but using q_i and $\overline{q}_i$ rather than q and r), as well as m sets of 3 rules, where each rule in the jth set concludes a literal c_j given an appropriate assignment for the "base" q_i literals. We then define the set of m queries, each insisting that one of the c_j literals must be entailed. See the appendix for the remaining details.

The proof for $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon^{Ord-Rules})$ is similar, but instead uses T_{ab} from Equation 4. Observe that the first answer returned to the `aORb(Z)` query depends on the "assignment" to the variable "a" (resp., "b") which depends on the order of the `a(0)` and `a(1)` clauses (resp., the order of the `b(0)` and `b(1)` clauses). To reduce a 3SAT problem, we need only define `or3` (for disjunction of 3 literals), and add queries that insist that each "clause" $c_i(X)$ have, as its first answer, `Yes`$[X = 1]$. (Again, the details appear in the appendix.) □

This theorem means that, unless $P = NP$, no polynomial-time algorithm can find an ordering of a list of impure proposition Horn clauses (resp., of a list of pure predicate calculus Horn clauses) that returns the correct answer (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; recall $\Upsilon_K^{Ord-Rules}(\mathrm{T})$ is the set of theories formed by applying a sequence of at most $K = K(|\mathrm{T}|)$ such individual moves. As a simple example, notice

$$\Upsilon_1^{Ord-Rules}(\{a,b,c,d\}) = \left\{ \begin{array}{ccc} \{b,a,c,d\} & \{b,c,a,d\} & \{b,c,d,a\} \\ \{a,b,d,c\} & \{a,c,b,d\} & \{a,c,d,b\} \\ \{c,a,b,d\} & \{d,a,b,c\} & \{a,d,b,c\} \end{array} \right\}$$

includes only the singly-modified theories, and so includes 9 of the $4! = 24$ possible permutations.

If K is constant, then we can trivially enumerate and test all $O(|\mathrm{T}|^K)$ theories in $\Upsilon_K^{Ord-Rules}[\mathrm{T}]$, and so the obvious decision problem becomes trivial:

Observation 1 *For constant K, the* $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon_K^{Ord-Rules}\,)$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon_K^{Ord-Rules}\,)$ *decision problems can each be solved in polynomial time.*

However, for larger $K(\cdot)$, the task again become intractable:

Theorem 3
For some $K(\mathrm{T}) = \Omega(\sqrt{|\mathrm{T}|})$*, each of* $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon_K^{Ord-Rules}\,)$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon_K^{Ord-Rules}\,)$ *is NP-complete.*

(These proofs uses the same basic "tricks" shown above, but deal with the NP-hard decision problem "Exact Cover by 3-Sets".)

These negative results show the intractability of the obvious proposal of using a breath-first traversal 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 $\mathrm{T}_1 \in \Upsilon_1^{Ord-Rules}[\mathrm{T}_0]$. If there are none, next consider all theories in $\Upsilon_2^{Ord-Rules}[\mathrm{T}_0]$ (formed by applying pairs of moves), and return any perfect $\mathrm{T}_2 \in \Upsilon_2^{Ord-Rules}[\mathrm{T}_0]$; and so forth.

Approximatability: Many decision problems correspond immediately to optimization problems; for example, the IndSet decision problem

> Given a graph $G = \langle N, E\rangle$ and a positive integer K, is there an independent set of size K — *i.e.*, a subset $S \subset N$ of at least $|S| \geq K$ nodes that are not connected to one another (*i.e.*, such that $\forall s_1, s_2 \in S, \langle s_1, s_2\rangle \notin E$) [24, p194]?

corresponds to the obvious maximization problem:

Definition 2 (MaxIndSet Maximalization Problem)
Given a graph $G = \langle N, E\rangle$, find the largest independent subset of N.

We can similarly identify the $\mathrm{DP}(\,\Upsilon^{Ord-Rules}\,)$ decision problem with the "$\mathrm{MAX}(\,\Upsilon^{Ord-Rules}\,)$" maximization problem: "Find the $\mathrm{T}^* \in \Upsilon^{Ord-Rules}(\mathrm{T})$ whose accuracy is maximal".

Now consider any algorithm $B(\cdot)$ that, given any $\mathrm{MAX}(\,\Upsilon^{Ord-Rules}\,)$ 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 $B(\langle \mathrm{T},\, S\,\rangle) \in \Upsilon^{Ord-Rules}(\mathrm{T})$. Then B's "performance ratio for the instance x" is defined as

$$MaxPerf(\; B,\, x\;) \;=\; MaxPerf_{\Upsilon^{Ord-Rules}}(\; B,\, x\;) \;=\; \frac{A(\,opt(x)\,)}{A(\,B(x)\,)}$$

where $opt(x) = opt_{MAX(\Upsilon^{Ord-Rules})}(x)$ is the optimal solution for this instance; *i.e.*, $opt(\langle \mathrm{T},\, S\rangle)$ is the theory $\mathrm{T}_{opt} \in \Upsilon^{Ord-Rules}(\mathrm{T})$ 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

$$\forall \text{instances } x \;\in\; \mathrm{MAX}(\,\Upsilon^{Ord-Rules}), \quad MaxPerf(\,B, x\,) \;\le\; g(|x|)$$

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 $\mathrm{MAX}(\,\Upsilon^{Ord-Rules})$ instances x.

Now let $Poly(\,\mathrm{MAX}(\,\Upsilon^{Ord-Rules}))$ be the collection of all *polynomial-time* algorithms that return legal answers to $\mathrm{MAX}(\,\Upsilon^{Ord-Rules})$ instances. It is natural to ask for the algorithm in $Poly(\,\mathrm{MAX}(\,\Upsilon^{Ord-Rules}))$ 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 $1(x) \equiv 1$ for $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon^{Ord-Rules})$ then a polynomial-time algorithm could produce the optimal solution to any $\mathrm{MAX}(\,\Upsilon^{Ord-Rules})$ instance; as $\mathrm{DP}_{Opt,Imp,Prop}(\,\Upsilon^{Ord-Rules})$ 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 function $c(x) \equiv c \in \Re^{+}$, then we could efficiently obtain a solution within a factor of c of optimal, which may be good enough for some applications.[7]

However, not all problems can be approximated. Following [12, 30], we define

Definition 3 *A maximization problem* MAX *is* POLYAPPROX *iff*

$$\forall \gamma \in \Re^{+},\; \exists B_{\gamma} \in \mathrm{Poly}(\,\mathrm{MAX}),\; \forall x \in \mathrm{MAX},\; MaxPerf_{\mathrm{MAX}}(\,B_{\gamma}, x\,) \;<\; |x|^{\gamma}\;.$$

Arora *et al.* [2] prove that

Theorem 4 (from [2]) *Unless $P = NP$, the "*MAXINDSET *maximization problem" is not* POLYAPPROX *— i.e., there is a $\gamma \in \Re^{+}$ such that no polynomial-time algorithm can produce a solution to arbitrary* MAXINDSET *problems to within K^{γ}, where K is the number of nodes in the graph.*

We use that result to prove:

Theorem 5 *Unless $P = NP$, neither $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon^{Ord-Rules})$ nor $\mathrm{MAX}_{Opt,Pur,PC1}(\,\Upsilon^{Ord-Rules})$ is* POLYAPPROX.

[7] 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 within a factor of 11/9 for any MAXBINPACKING maximization problem; see [24, Theorem 6.2].

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

4.2 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 $\Upsilon^{Ord-Antes}(\mathrm{T})$ be the set of theories obtained by reordering the antecedents in T's rules, and ask the same questions asked above: sample complexity, computational complexity and approximatability. Here, we obtain the same results, *mutatis mutandis*:

First, note that $|\Upsilon^{Ord-Antes}(\mathrm{T})| = \prod_{c \in T}(\#\mathrm{Antes(c)})! = O(|\mathrm{T}|^{|T|})$, where $\#\mathrm{Antes(c)} \in \mathcal{Z}^{\geq 0}$ is the number of antecedents in the clause c. Using Theorem 1, this means we need only a polynomial number of samples.

Addressing the computational complexity of these tasks, we see

Theorem 6
Each of $\mathrm{DP}_{Perf,Imp,Prop}(\ \Upsilon^{Ord-Antes}\ )$, $\mathrm{DP}_{Perf,Pur,PC1}(\ \Upsilon^{Ord-Antes}\ )$, $\mathrm{DP}_{Perf,Imp,Prop}(\ \Upsilon_K^{Ord-Antes}\ )$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\ \Upsilon_K^{Ord-Antes}\ )$ *is NP-complete.*

(Notice this includes both the limited $\Upsilon_K^{Ord-Antes}$ and unlimited $\Upsilon^{Ord-Antes}$ transformations.)

Proof (sketch): The proof for $\mathrm{DP}_{Perf,Imp,Prop}(\ \Upsilon^{Ord-Antes}\ )$ (resp., $\mathrm{DP}_{Perf,Imp,Prop}(\ \Upsilon_K^{Ord-Antes}\ )$) resembles the proof of Theorem 2 (resp., Theorem 3) but uses the observation that reordering the antecedents of "`q :- !, fail.`" (within the theory $\langle \ldots,$ `q :- !, fail.`, `q.`, $\ldots\rangle$) to form "`q :- fail, !.`" has the effect of allowing `q` to be entailed. To deal with $\mathrm{DP}_{Perf,Pur,PC1}(\ \Upsilon^{Ord-Antes}\ )$ and $\mathrm{DP}_{Perf,Pur,PC1}(\ \Upsilon_K^{Ord-Antes}\ )$, replace each "$\mathtt{a}_j$`(0).`" and "$\mathtt{a}_j$`(1).`" pair with the single clause

$$\mathtt{a}_j\mathtt{(Y)\ :\!-\ prefer0(Y),\ prefer1(Y).} \tag{5}$$

and also include the four atomic clauses

$$\mathtt{prefer0(0).\quad prefer0(1).\qquad prefer1(1).\quad prefer1(0).} \tag{6}$$

in this order. If we use Equation 5, we see $\mathtt{a}_j$`(Y)` will first return `Yes`$[Y\!=\!0]$; but we can get `Yes`$[Y\!=\!1]$ by simply inverting the order of Equation 5's antecedents. Thus, by reordering the antecedents, we can again arbitrarily set the first answer to the various subqueries, and thereby determine the first answer to the top-level query. □

We can use this same basic "proof-to-proof transformation" to transform the proof of Theorem 5 to show that:

Theorem 7
Unless $P = NP$, neither $\mathrm{MAX}_{Opt,Imp,Prop}(\ \Upsilon^{Ord-Antes}\ )$ *nor* $\mathrm{MAX}_{Opt,Pur,PC1}(\ \Upsilon^{Ord-Antes}\ )$ *is* PolyApprox.

4.3 Adding or Deleting Clauses

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

Theorem 8 (from [28])
In the pure context, for each $\Upsilon \in \{\Upsilon^{Add-Rules},\ \Upsilon^{Del-Rules}\}$

- $\mathrm{DP}_{Perf,Pur,Prop}(\,\Upsilon\,)$ *can be solved in polynomial time*
- *Each of* $\mathrm{DP}_{Opt,Pur,Prop}(\,\Upsilon\,)$ *and* $\mathrm{DP}_{Opt,Pur,\ PC-All}(\,\Upsilon\,)$ *is NP-hard, but is trivial to approximate:*
  $\exists B_\Upsilon \in \mathrm{Poly}(\ \mathrm{MAX}_{Opt,Pur,\rho}(\,\Upsilon\,)\ ),$
  $\forall x \in \mathrm{MAX}_{Opt,Pur,\rho}(\,\Upsilon\,),\ \mathit{MaxPerf}_{MAX_{Opt,Pur,\rho}(\Upsilon)}(\,B_\Upsilon, x\,)\ \le\ 2.$
  for $\rho =$ *"Prop" or* $\rho =$ *"PC-All".*

(Notice Theorem 8 considers the pure "PC-All" context, which seeks *all* answers to each query, rather than the impure "PC1", which seeks only the first answer.)

Each of these *pure* maximization problems is trivially approximated, at worst within a factor of 2. However, in the impure setting, these tasks are more difficult. To be precise, we first specify that the $\Upsilon^{Add-Rules}$ operators add rules to the *end* of the theory. (Otherwise, the predicate calculus tasks remain trivial.)

Theorem 9
For each $\Upsilon \in \{\ \Upsilon^{Add-Rules},\ \Upsilon^{Del-Rules}\ \}$,

(1) *Each of* $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon\,)$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon\,)$ *is NP-hard, and*
(2) *unless* $P = NP$*, neither* $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon\,)$ *nor* $\mathrm{MAX}_{Opt,Pur,PC1}(\,\Upsilon\,)$ *is* PolyApprox.

(Note that $\mathrm{DP}_{Perf,Imp,PC1}(\,\Upsilon^{Add-Rules}\,)$ is not in NP: Given function symbols, $\Upsilon^{Add-Rules}(\mathrm{T})$ can contain an unbounded number of possible theories.)

Proof (sketch): All three $\Upsilon^{Del-Rules}$ claims follow from some earlier theorem merely by noting that deleting a "`a :- !, fail.`" clause (resp., "`a(0).`") from a theory that later includes "`a.`" (resp., "`a(1).`") causes `a` to be entailed (resp., `a(1)` to be found first). The proofs for the $\Upsilon^{Add-Rules}$-claims all require different tricks, which often require queries that specify that some literal must *not* be entailed. See the appendix. □

It is worth noting that all four of our $\Upsilon^{Add-Rules}$ results hold even if we consider only adding *atomic* clauses; in fact, these added clauses are always ground symbols. This further distinguishes our results from ILP's, where the added clauses can be arbitrary.

To address the sample complexity issue, notice that $\ln(|\Upsilon^{Del-Rules}|) = |\mathrm{T}|$, which means a polynomial number of samples is sufficient to make the familiar PAC-style guarantees. Similarly, $\ln(|\Upsilon^{Add-Rules}|)$ is polynomial in the size of the theory and the language $\mathcal{L}$, in the propositional case. In the predicate calculus case, however, $\Upsilon^{Add-Rules}$ can potentially be arbitrarily large, meaning the above analysis does not apply. (Note, however, that our

		Order Rules	Order Antes	Add Rules	Delete Rules
Prop'n	Pure	(no effect)*	(no effect)*	DP: trivial† MAX: NP	DP: trivial† MAX: NP
	Impure	DP: NP MAX: ¬PA	DP: NP MAX: ¬PA	DP: NP MAX: ¬PA	DP: NP MAX: ¬PA
PC-All	Pure	(no effect)*	(no effect)*	DP: NP MAX: ≤ 2‡	DP: NP MAX: ≤ 2‡
PC-1	Pure	DP: NP MAX: ¬PA	DP: NP MAX: ¬PA	DP: NP MAX: ¬PA	DP: NP MAX: ¬PA

Legend:
DP = Decision problem of finding *perfect* theory
MAX = Optimizing problem of finding best theory in general
NP = "NP-hard"
¬PA = "Not poly approx"
*: Trivial to find best, as reordering has no effect.
†: Trivial when queries are atomic. If queries are "disjunctions", task is NP-hard [28] .
‡: "≤ 2" means "Can be approximated to within factor of 2"

(Hardness/non-approximatability of "*impure* PC-All/PC-1 tasks" follows immediately from hardness/non-approximatability of the simpler impure Propositional tasks.)

Table 2: Summary of Computational Complexity/Approximatability Results

negative results that deal with the computational hardness of these tasks all involve simpler additions, and hold in "function-free" theories.)

It is easy to show that these same claims also apply to the tasks of adding or deleting *antecedents*: In the pure context, it is trivial to determine whether one can form a perfect theory by adding or deleting antecedent in the propositional case, but these tasks become NP-hard in the impure case. In terms of finding the optimal theory in space of adding (resp., deleting) antecedents: This task is (NP-hard but) easily approximatable in *pure* contexts, but is not PolyApprox in *impure* contexts. (These proofs are isomorphic to the ones appearing in the appendix.)

5 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 [44, 38, 40]. 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 (resp., ordering of the antecedents) is hopelessly intractable: no efficient algorithm can produce even a good approximation to the optimum. This resonates with earlier analyses of the theory revision task, and justifies the standard approach of hill-climbing to a locally-optimal theory. Finally, we 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. These results are summarized in Table 2.

A Proofs

This appendix explicitly proves that each NP-complete task is NP-hard; in each case, it is trivial to see that the problem is in NP.

Theorem 2 *Each of* $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Rules})$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon^{Ord-Rules})$ *is NP-complete.*

Proof: We reduce the canonical NP-complete task 3SAT to our problems:

> **Definition 4 (3SAT Decision Problem,** from [24, p259]**:)** *Given a set* $U = \{u_1, \ldots, u_n\}$ *of variables and formula* $\varphi = \{c_1, \ldots, c_m\}$ *(a conjunction of clauses over* U*) such that each clause* $c \in \varphi$ *is a disjunction of* 3 *(positive or negative) literals, is there a satisfying truth assignment for* φ*?*

We first deal with $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Rules})$: Given any 3SAT formula $\varphi = \{c_1, c_2, \cdots, c_m\}$ over the variables $U = \{u_1, \ldots, u_n\}$, use the following $3n + 3m$-clause theory

$$\mathrm{T}_{\varphi}^{(Prop)} \quad = \quad \left\{ \begin{array}{lll} \left.\begin{array}{l} \mathtt{u}_i \ \mathtt{:-\ !,\ fail.} \\ \mathtt{u}_i\mathtt{.} \\ \bar{\mathtt{u}}_i \ \mathtt{:-\ not(\ } \mathtt{u}_i \mathtt{\ ).} \end{array}\right\} & \text{for } u_i \in U \\ \mathtt{c}_j \ \mathtt{:-\ } \mathtt{u}_i\mathtt{.} & \text{if } u_i \in c_j \\ \mathtt{c}_j \ \mathtt{:-\ } \bar{\mathtt{u}}_i\mathtt{.} & \text{if } \bar{u}_i \in c_j \end{array} \right\} \tag{7}$$

and let $S_{\varphi}^{(Prop)}$ be the following m query/answer pairs:

$$S_{\varphi}^{(Prop)} \quad = \quad \left\{ \ \langle \mathtt{c}_j;\ \mathtt{Yes} \rangle \quad \text{for } c_j \in \varphi \ \right\}$$

(Of course, each $\mathtt{u}_i$ corresponds to the u_i positive literal, $\bar{\mathtt{u}}_i$ to the $\bar{u}_i$ negative literal, and $\mathtt{c}_j$ to the j^{th} clause c_j.)

We need only show that there is a theory $\mathrm{T}_{opt} \in \Upsilon^{Ord-Rules}[\mathrm{T}_{\varphi}^{(Prop)}]$ whose accuracy is $\mathrm{A}(\mathrm{T}_{opt}) = 1$ iff there is a satisfying assignment of φ.

This is straightforward: The only re-orderings that matter concern the relative positions of the "$\mathtt{u}_i$ `:- !, fail.`" and "$\mathtt{u}_i$`.`" clauses. In the order shown in Equation 7, the theory entails $\bar{\mathtt{u}}_i$ but not $\mathtt{u}_i$; if reversed, then it entails $\mathtt{u}_i$ but not $\bar{\mathtt{u}}_i$. In either case, it entails exactly one of $\{\mathtt{u}_i,\ \bar{\mathtt{u}}_i\}$, and so corresponds immediately to a legal assignment. Notice further that the resulting theory entails each $\mathtt{c}_j$ iff the associated assignment satisfies c_j, which means φ has a satisfying assignment iff there is an ordering which answers `Yes` to each $\mathtt{c}_j$, which means the ordering is perfect.

The proof for $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon^{Ord-Rules}\,)$, in essence, replaces each $\mathtt{u}_i$ in $\mathrm{T}_\varphi^{(Prop)}$ with $\mathtt{u}_i$`(1)`, and each $\bar{\mathtt{u}}_i$ with $\mathtt{u}_i$`(0)`: Here, to simplify the description, we use the MONOTONE3SAT problem, which is the NP-complete specialization of 3SAT in which each clause includes either only positive literals, or only negative literals [24, p259]. Let P be the subset of clauses whose elements are of the form $c_j = \{u_{j1}, u_{j2}, u_{j3}\}$ and N be the subset whose elements are of the form $c_j = \{\bar{u}_{j1}, \bar{u}_{j2}, \bar{u}_{j3}\}$. Then let $\mathrm{T}_\varphi^{(PC)}$ be

```
{ u_i(0).                                             for u_i ∈ U
  u_i(1).                                             for u_i ∈ U
  c_j(X) :- u_j1(V1), u_j2(V2), u_j3(V3),             for c_j = {u_j1, u_j2, u_j3} ∈ P
            or3(V1, V2, V3, X).
  c_j(X) :- u_j1(V1), u_j2(V2), u_j3(V3),             for c_j = {ū_j1, ū_j2, ū_j3} ∈ N
            nand3(V1, V2, V3, X).
  or3(0,0,0,0).     or3(0,0,1,1).     or3(0,1,0,1).     or3(0,1,1,1).
  or3(1,0,0,1).     or3(1,0,1,1).     or3(1,1,0,1).     or3(1,1,1,1).
  nand3(0,0,0,1).   nand3(0,0,1,1).   nand3(0,1,0,1).   nand3(0,1,1,1).
  nand3(1,0,0,1).   nand3(1,0,1,1).   nand3(1,1,0,1).   nand3(1,1,1,0).  }
```

(where each $\mathtt{u}_i$`(0)` appears before the corresponding $\mathtt{u}_i$`(1)`) and let $S_\varphi^{(PC)}$ be the m query/answer pairs:

$$S_\varphi^{(PC)} \quad = \quad \left\{ \ \langle \mathtt{c}_j(X);\ \mathtt{Yes}[X\!=\!1] \rangle \quad \text{for } c_j \in \varphi \ \right\}$$

The `or3` predicate "returns" the disjunction of its first three arguments, viewing 1 as true and 0 as false, and the `nand3` predicate "returns" the disjunction of the *negation* of its first three arguments.

Clearly there is a satisfying assignment of φ iff there is a theory $\mathrm{T}_{opt} \in \Upsilon^{Ord-Rules}[\mathrm{T}_\varphi^{(PC)}]$, with a particular ordering of the $\mathtt{u}_i$`(0)` and $\mathtt{u}_i$`(1)` clauses, whose accuracy is $\mathrm{A}(\,\mathrm{T}_{opt}\,) = 1$.
□ (Theorem 2)

Theorem 3 *For some* $K(\mathrm{T}) = \Omega(\sqrt{|\mathrm{T}|})$*, each of* $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon_K^{Ord-Rules}\,)$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon_K^{Ord-Rules}\,)$ *is NP-complete.*

Proof: These proofs use the NP-complete problem

> **Definition 5** (X3C **[Exact Cover by 3-Sets], from [GJ79, p221]**) *Given a set* X *with* $|X| = 3k$ *elements and a collection* C *of 3-element subsets of* X*, does* C *contain an* exact cover *for* X*; i.e., a subcollection* $C' \subseteq C$ *such that every element of* X *occurs in exactly one member of* C' *?*

To deal with $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon_K^{Ord-Rules}\,)$, let

$$\mathrm{T}_{XC}^{(Prop)} \quad = \quad \left\{ \begin{array}{ll} \texttt{x}_i \texttt{ :- c}_j\texttt{.} & \text{when } x_i \in c_j \\ \left.\begin{array}{l} \texttt{c}_j \texttt{ :- !, fail.} \\ \texttt{c}_j\texttt{.} \end{array}\right\} & \text{for } c_j \in C \end{array} \right\}$$

let $S_{XC}^{(Prop)}$ be the $3k$ query/answer pairs

$$S_{XC}^{(Prop)} \quad = \quad \left\{ \; \langle \texttt{x}_i;\ \texttt{Yes} \rangle \quad \text{for } x_i \in X \; \right\}$$

and let $K = k$.

Our task is to re-order at most k clauses, to obtain a perfect theory. By inspection, we need only consider the relative ordering of the "$\texttt{c}_j$ `:- !, fail.`" and "$\texttt{c}_j$`.`" clauses. If there is an exact covering, say $\{c_1, \ldots, c_k\}$, then we can form a perfect theory by reordering the clauses for the corresponding $\texttt{c}_j$s; and vice versa.

Notice also that $|\mathrm{T}_{XC}^{(Prop)}| = \sum_{x_i \in X} 3 \times 3 + 4|C| \;\leq\; 9 \times 3k + 4 \times (|X|^2/2) \;=\; 27k + 18k^2 \;=\; O(k^2)$, which means $K(\mathrm{T}) = k = \Omega(\sqrt{|\mathrm{T}|})$ is sufficient.

To handle $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon_K^{Ord-Rules}\,)$, we use the following theory, $\mathrm{T}_{XC}^{(PC)}$:

$$\left\{ \begin{array}{ll} \texttt{x}_i^{(j)}\texttt{(Z}_j\texttt{) :- x}_i^{(j-1)}\texttt{(Z}_{j-1}\texttt{), c}_{ij}\texttt{(Y}_j\texttt{), or2(Y}_j\texttt{, Z}_{j-1}\texttt{, Z}_j\texttt{).} & \text{when } x_i \in c_{ij} \texttt{ for each } j \\ \left.\begin{array}{l} \texttt{c}_j\texttt{(0).} \\ \texttt{c}_j\texttt{(1).} \end{array}\right\} & \text{for } c_j \in C \\ \texttt{or2(0, 0, 0). or2(0, 1, 1). or2(1, 0, 1). or2(1, 1, 1).} & \end{array} \right\}$$

and

$$S_{\varphi}^{(Prop)} \quad = \quad \left\{ \; \langle \texttt{x}_i^{(\ell_i)}(Y),\ \texttt{Yes}[Y\!=\!1] \rangle \quad \text{for } x_i \in X \; \right\}$$

and $K = k$. To explain the notation: Each x_i element is a member of the $\ell_i \leq |C|$ sets $c_{i1}, c_{i2}, \ldots, c_{i\ell_i} \in C$; hence, there are ℓ_i clauses associated with x_i, headed by $\texttt{x}_i^{(1)}, \ldots, \texttt{x}_i^{(\ell_i)}$. (The $\texttt{x}_i^{(1)}$-headed clause is the degenerate "$\texttt{x}_i^{(1)}\texttt{(Z}_1\texttt{) :- c}_{i1}\texttt{(Z}_1\texttt{).}$".) The `or2` predicate "returns" the disjunction of its first two arguments.

Hence, the first answer returned to the $\texttt{x}_i^{(\ell_i)}\texttt{(Y)}$ query will be $Y\!=\!1$ only if, for at least one of the associated classes, say "$\texttt{c}_{i\chi}$", the "$\texttt{c}_{i\chi}\texttt{(Y}_\chi\texttt{)}$" subquery returns $Y_\chi\!=\!1$, which happens only if the "$\texttt{c}_{i\chi}\texttt{(1).}$" atomic clause is moved before "$\texttt{c}_{i\chi}\texttt{(0).}$". Hence, once again, we can find a perfect theory iff we can re-order exactly $K = k$ of the "$\texttt{c}_j\texttt{(0).}$" and "$\texttt{c}_j\texttt{(1).}$" clauses.

□ (Theorem 3)

Theorem 5 *Unless $P = NP$, neither* $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon^{Ord-Rules}\,)$ *nor* $\mathrm{MAX}_{Opt,Pur,PC1}(\,\Upsilon^{Ord-Rules}\,)$ *is* PolyApprox.

Proof: Based on Theorem 4, we reduce the "not-PolyApprox–hard" MaxIndSet maximalization problem (Definition 2) to these problems: We first deal with $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon^{Ord-Rules}\,)$. Given any graph $G = \langle N, E \rangle$, let $\mathrm{T}_G^{(Prop)}$ be the following $3|N| + |E|$ propositions (requiring

$15|N| + 6|E|$ symbols)

$$\mathrm{T}_G^{(Prop)} \quad = \quad \left\{ \begin{array}{ll} \left.\begin{array}{l} \texttt{n}_j\,\texttt{.} \\ \texttt{n}_j\ \texttt{:- !, fail.} \\ \texttt{good}_j\ \texttt{:- not(bad), n}_j\,\texttt{.} \end{array}\right\} & \text{for } \ n_j \in N \\ \texttt{bad :- n}_{i_1}\texttt{, n}_{i_2}\texttt{.} & \text{for } \ e_i = \langle \texttt{n}_{i_1}, \texttt{n}_{i_2} \rangle \in E \end{array} \right\}$$

and

$$S_G^{(Prop)} \quad = \quad \{ \langle \texttt{good}_j;\ \texttt{Yes} \rangle \quad \text{for } j = 1..N \ \}$$

To derive any $\texttt{good}_j$ literal, the `bad` subquery must fail, which means, for each $e_i = \langle \texttt{n}_{i_1}, \texttt{n}_{i_2} \rangle$, at least one of $\texttt{n}_{i_1}$ or $\texttt{n}_{i_2}$ must not be derivable. This can only happen if we exchange the order of the (say) "$\texttt{n}_{i_1}$" and "$\texttt{n}_{i_1}$ `:- !, fail`" clauses.

For notation, let R represent the set of $\texttt{n}_j$ literals that are *not* switched; notice here that $\texttt{good}_j$ is entailed. As R can contain at most one node from each edge, it is an independent set.

Now observe that the $\texttt{good}_j$ query can only contribute its $\frac{1}{|N|}$ to the program's accuracy score if the $\texttt{n}_j$ literal is derivable, that is, if it has *not* been switched; *i.e.*, if it is a member of R. Hence, the score for this program is $\frac{|R|}{|N|}$.

Now suppose, for every $\epsilon \in \Re^+$, there is a polynomial-time algorithm $B_\epsilon(\cdot)$ such that, for any theory T and query-set S, $B_\epsilon(\mathrm{T})$ returns a theory $\mathrm{T}_\epsilon \in \Upsilon^{Ord-Rules}(\mathrm{T})$ whose accuracy is within a factor of $|\langle \mathrm{T}, S\rangle|^\epsilon$ of the accuracy of the optimal $opt(\mathrm{T}) \in \Upsilon^{Ord-Rules}(\mathrm{T})$; *i.e.*, such that $\frac{A(\,opt(T)\,)}{A(B_\epsilon(T))} \leq |\langle \mathrm{T}, S\rangle|^\epsilon$. We could then use these algorithms to find approximately optimal solutions to any MaxIndSet problem, as follows:

Given any MaxIndSet problem $G = \langle N, E\rangle$ (with $|N| \geq 9$), use the above transformation to form the $\mathrm{T}_G^{(Prop)}$ theory and $S_G^{(Prop)}$ queries. Let $R^* \in \mathcal{Z}^+$ be the optimal solution to G (*i.e.*, the maximal number of independent nodes); this corresponds to the optimal solution for $\langle \mathrm{T}_G^{(Prop)}, S_G^{(Prop)}\rangle$, call it $\mathrm{T}_{G,opt}$, whose accuracy is $\mathrm{A}(\,\mathrm{T}_{G,opt}\,) = \frac{R^*}{K}$. Now use the $B_{\gamma/3}$ algorithm to produce a theory $\mathrm{T}_{G,\gamma/3}$ whose accuracy $\mathrm{A}(\,\mathrm{T}_{G,\gamma/3}\,) = \frac{R_{\gamma/3}}{K}$ satisfies the performance ratio $\frac{A(T_{G,opt})}{A(T_{G,\gamma/3})} = \frac{R^*}{K} / \frac{R_{\gamma/3}}{K} = \frac{R^*}{R_{\gamma/3}} \leq |\langle \mathrm{T}_G, S_G^{(Prop)}\rangle|^{\gamma/3} \leq (15|N| + 6|E| + 2|N|)^{\gamma/3}$. Notice this corresponds to a feasible MaxIndSet solution to G with $R_{\gamma/3}$ nodes. As $|E| \leq |N|^2$ and $|N| \geq 9$, $(17|N| + 6|E|)^{\gamma/3} \leq (|N|^3)^{\gamma/3} = |N|^\gamma$, meaning we have produced a solution (to G) with a performance ratio of under $|N|^\gamma$ in polynomial time. As this γ can be arbitrarily small, this contradicts Theorem 4, assuming $P \neq NP$.

The proof for $\mathrm{MAX}_{Opt,Pur,PC1}(\,\Upsilon^{Ord-Rules}\,)$ resembles the above proof, but is more cumbersome: Here, given any graph $G = \langle N, E\rangle$, form the $3|N| + |E| + 8$-clause theory $\mathrm{T}_G^{(PC)}$:

$$
\left\{
\begin{array}{ll}
\left.\begin{array}{l}
\texttt{n}_j\texttt{(1).} \\
\texttt{n}_j\texttt{(0).} \\
\texttt{good}_j\texttt{(OK, IO) :- bad}_{|E|}\texttt{(OK), n}_j\texttt{(IO).}
\end{array}\right\} & \texttt{for } n_j \in N \\
\left.\begin{array}{l}
\texttt{bad}_i\texttt{(OK) :- n}_{i_1}\texttt{(IO}_a\texttt{), n}_{i_2}\texttt{(IO}_b\texttt{), and2(IO}_a\texttt{, IO}_b\texttt{, OK}_i\texttt{),} \\
\qquad\qquad \texttt{bad}_{i-1}\texttt{(OK}_{i-1}\texttt{), or2(OK}_i\texttt{, OK}_{i-1}\texttt{, OK).}
\end{array}\right\} & \texttt{for } \langle \texttt{n}_{i_1}, \texttt{n}_{i_2}\rangle \in E \\
\texttt{and2(1, 1, 1).} \quad \texttt{or2(1, 1, 1).} & \\
\texttt{and2(0, 1, 0).} \quad \texttt{or2(0, 1, 1).} & \\
\texttt{and2(1, 0, 0).} \quad \texttt{or2(1, 0, 1).} & \\
\texttt{and2(0, 0, 0).} \quad \texttt{or2(0, 0, 0).} &
\end{array}
\right\}
$$

where the body of the $\texttt{bad}_1$ clause includes only the first 3 literals:

$$\texttt{bad}_1\texttt{(OK) :- n}_{1a}\texttt{(IO}_a\texttt{), n}_{1b}\texttt{(IO}_b\texttt{), and2(IO}_a\texttt{, IO}_b\texttt{, OK) .}$$

The 8 clauses defining the `or2` and `and2` predicates mean that `and2(a,b,c)` holds iff $\texttt{c} = \texttt{a}\&\texttt{b}$, and `or2(a,b,c)` holds iff $\texttt{c} = \texttt{a} \vee \texttt{b}$.

The $K = |N|$ queries are

$$\{\ \langle \texttt{good}_j\texttt{(OK, IO)};\ \texttt{Yes}[\texttt{OK}=\texttt{0},\ \texttt{IO}=\texttt{1}]\rangle \quad \text{for each} \ \ \texttt{n}_j \in N\ \}$$

By inspection, the only rule-reordering that can affect accuracy is moving a "$\texttt{n}_j\texttt{(0)}$" clause relative to the corresponding "$\texttt{n}_j\texttt{(1)}$" clause. Let R include n_j for each $\texttt{n}_j\texttt{(1)}$ clause that remains before the corresponding $\texttt{n}_j\texttt{(0)}$.

To derive the proper binding for each $\texttt{good}_j\texttt{(OK, IO)}$ query, the first answer to the $\texttt{bad}_{|E|}\texttt{(OK)}$ query must be $\texttt{Yes}[\texttt{OK}=0]$. Using a simple inductive argument, this requires, for each $e_i = \langle \texttt{n}_{i_1}, \texttt{n}_{i_2}\rangle$, that either the first binding to $\texttt{IO}_a$ returned for $\texttt{n}_{i_1}\texttt{(IO}_a\texttt{)}$ be $\texttt{Yes}[\texttt{IO}_a=0]$, or the first binding to $\texttt{IO}_b$ returned for $\texttt{n}_{i_2}\texttt{(IO}_b\texttt{)}$ be $\texttt{Yes}[\texttt{IO}_b=0]$. This means that at least one of $\texttt{n}_{i_1}\texttt{(0)}$ or $\texttt{n}_{i_2}\texttt{(0)}$ must be ordered before the corresponding $\texttt{n}_{i_1}\texttt{(1)}$ (resp., $\texttt{n}_{i_2}\texttt{(1)}$) clause. Hence, the set R can contain at most one node of each arc, meaning it is an independent set.

The `IO` variable of the $\texttt{good}_j\texttt{(OK, IO)}$ query will only be bound correctly to $\texttt{IO}=\texttt{1}$ if the corresponding $\texttt{n}_j\texttt{(1)}$ literal appears before $\texttt{n}_j\texttt{(0)}$; *i.e.*, if $n_j \in R$. Hence, a program can have an accuracy score of $\frac{|R|}{K}$ if R corresponds to an independent set in G, and an accuracy score of 0 otherwise.

(The rest of this proof is essentially identical to the one above.)

□ (Theorem 5)

Theorem 6 *Each of* $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Antes})$, $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon^{Ord-Antes})$, $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon_K^{Ord-Antes})$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon_K^{Ord-Antes})$ *is NP-complete.*

Proof: The proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Antes})$ is essentially the same as the proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon^{Ord-Rules})$ (Theorem 2), using the observation that reordering the antecedents of "`u :- !, fail.`" to form "`u :- fail, !.`" has the effect of allowing `u` to be entailed. The proof for $\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon_K^{Ord-Antes})$ is similarly related to the proof for

$\mathrm{DP}_{Perf,Imp,Prop}(\Upsilon_K^{Ord-Rules})$ (Theorem 3), as changing "`c`$_j$ `:- !, fail.`" to "`c`$_j$ `:- fail, !.`" causes `c`$_j$ to be entailed.

For $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon^{Ord-Antes})$, replace each of (Theorem 2) $T_\varphi^{(PC)}$'s "`u`$_j$`(0).`" and "`u`$_j$`(1).`" pair of clauses with the single clause "`u`$_j$`(Y) :- prefer0(Y), prefer1(Y).`", and also include the four atomic "`prefer`$i(j)$" clauses shown in Equation 6. Notice the first answer returned to the (sub)query "`u`$_j$`(Y)`" is $Y=0$, when using the initial "`u`$_j$`(Y) :- prefer0(Y), prefer1(Y).`" clause, but if we re-order the clause's antecedents to "`u`$_j$`(Y) :- prefer1(Y), prefer0(Y).`", we get $Y=1$. The rest of the proof is identical to the proof that $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon^{Ord-Rules})$ is NP-hard in Theorem 2.

The proof for $\mathrm{DP}_{Perf,Pur,PC1}(\Upsilon_K^{Ord-Antes})$ follows from the proof of Theorem 3, using this same trick of replacing each pair {`c`$_j$`(0).`, `c`$_j$`(1).`} with the single clause "`c`$_j$`(Y) :- prefer0(Y), prefer1(Y).`" and by including the four atomic clauses in Equation 6. As above, we can re-order the "`prefer0`" and "`prefer1`" literals of the "`c`$_j$`(Y) :- prefer0(Y), prefer1(Y).`" clauses to get different answers to the "`c`$_j$`(Y)`" subquery; etc. □ (Theorem 6)

Theorem 7 *Unless $P = NP$, neither* $\mathrm{MAX}_{Opt,Imp,Prop}(\Upsilon^{Ord-Antes})$ *nor* $\mathrm{MAX}_{Opt,Pur,PC1}(\Upsilon^{Ord-Antes})$ *is* PolyApprox.

Proof: To show that "$\mathrm{MAX}_{Opt,Imp,Prop}(\Upsilon^{Ord-Antes})$ is not PolyApprox", just modify Theorem 5's "$\mathrm{MAX}_{Opt,Imp,Prop}(\Upsilon^{Ord-Rules})$ is not PolyApprox" proof using the same $S_G^{(Prop)}$ queries but changing the initial theory to

$$\mathrm{T}_G^{(Prop)\prime} = \left\{ \begin{array}{ll} \left.\begin{array}{l} \texttt{n}_j\ \texttt{:- !, fail.} \\ \texttt{n}_j\texttt{.} \\ \texttt{good}_j\ \texttt{:- not(bad), n}_j\texttt{.} \end{array}\right\} & \text{for } n_j \in N \\ \texttt{bad :- n}_{i_1}\texttt{, n}_{i_2}\texttt{.} & \text{for } e_i = \langle \texttt{n}_{i_1}, \texttt{n}_{i_2} \rangle \in E \end{array} \right\}$$

(Notice we have inverted the order of the "`n`$_j$ `:- !, fail.`" and "`n`$_j$`.`" clauses.) Now observe that the only rules whose antecedent-order matters are the "`n`$_j$ `:- !, fail.`" rules. Here, by reordering those antecedents, we obtain the same effect as re-ordering this rule and the atomic `n`$_j$. (*I.e.*, here we re-use the same "theorem to theorem transformation" applied above to transform the proof of Theorem 2 to apply to Theorem 6.)

To show that "$\mathrm{MAX}_{Opt,Pur,PC1}(\Upsilon^{Ord-Antes})$ is not PolyApprox", modify the $\mathrm{T}_G^{(PC)}$ from Theorem 5 by replacing each "`n`$_j$`(1).`" and "`n`$_j$`(0).`" pair of rules with "`n`$_j$`(X) :- prefer1(X), prefer0(X).`", and adding the four atomic clauses in Equation 6. Now just replay the same proof of Theorem 5, replacing the "move `n`$_j$`(1)` before `n`$_j$`(0)`" with "reorder the antecedents of "`n`$_j$`(X) :- prefer1(X), prefer0(X).`".

Notice also that, due to the ordering of the `and2(...)` and `or2(...)` atomic clauses in the database, re-arranging the order of the antecedents of the `bad`$_i$ rules can only be detrimental: The only ordering that can lead to a different answer involve moving either the `and2` or `or2` literal to before some other literals. Consider first moving the `or2` literal forward, and notice that the only change this can produce is a binding that includes `OK` $=$ `1`, rather than `OK` $=$ `0` (*e.g.*, `or2(OK`$_i$`, 0, OK)` returns `Yes`[{`OK`$_i$=`1`, `OK`=`1`}], etc.); this is sufficient to insure that `bad`$_{|E|}$`(OK)` returns `OK` $=$ `1`, which again means the resulting theory will have an accuracy of 0. Similarly moving $\texttt{and2(IO}_a\texttt{, IO}_b\texttt{, OK}_i\texttt{)}$ to the first position will return $\texttt{Yes}[\{\texttt{IO}_a\!=\!\texttt{1}, \texttt{IO}_b\!=\!\texttt{1}, \texttt{OK}_i\!=\!\texttt{1}\}]$, which means the resulting theory will have 0 accuracy. If we move this literal to after the "$\texttt{n}_{i_1}\texttt{(IO}_a\texttt{)}$" antecedent, there are two cases to consider: If $\texttt{IO}_a$ is bound to 1, then the $\texttt{and2(1, IO}_b\texttt{, OK}_i\texttt{)}$ will match and2(1, 1, 1) and so bind $\texttt{OK}_i$ to 1, leading to the case mentioned above. Alternatively, if $\texttt{IO}_a$ is bound to 0, then this will bind $\texttt{OK}_i$ to 0, which is the appropriate answer here (as here we know that one of the antecedents has the 0 value). □ (Theorem 7)

Theorem 9
For each $\Upsilon \in \{\ \Upsilon^{Add-Rules},\ \Upsilon^{Del-Rules}\ \}$,
(1) *Each of* $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon\,)$ *and* $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon\,)$ *is NP-hard, and*
(2) *unless P = NP, neither* $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon\,)$ *nor* $\mathrm{MAX}_{Opt,Pur,PC1}(\,\Upsilon\,)$ *is* PolyApprox.

Proof: We first deal with the $\Upsilon^{Del-Rules}$ claims, each of which is a simple extension of an earlier theorem. To show that $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon^{Del-Rules}\,)$ (resp., $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon^{Del-Rules}\,)$) is NP-hard, just use the $\mathrm{T}_\varphi^{(Prop)}$ (resp., $\mathrm{T}_\varphi^{(PC)}$) theory from Theorem 2, and note that deleting the "$\texttt{u}_i$:- !, fail." clause causes $\texttt{u}_i$ to be entailed, and so has the same effect as moving "$\texttt{u}_i$:- !, fail." to after "$\texttt{u}_i$." (resp., deleting $\texttt{u}_i\texttt{(0)}$ means $\texttt{u}_i\texttt{(1)}$ will be first answer found, etc.) We can use the same idea to convert the proof of Theorem 5 to show the non-approximatability of $\mathrm{MAX}_{Opt,Imp,Prop}(\,\Upsilon^{Del-Rules}\,)$, as here deleting "$\texttt{n}_j$:- !, fail." from $\mathrm{T}_G^{(Prop)}$ produces a theory that entails $\texttt{n}_j$. For $\mathrm{MAX}_{Opt,Pur,PC1}(\,\Upsilon^{Del-Rules}\,)$, use the $\mathrm{T}_G^{(PC)}$ theory shown in Theorem 5, and notice that deleting any "$\texttt{n}_j\texttt{(1)}$." has the same effect as moving this $\texttt{n}_j\texttt{(1)}$ to after $\texttt{n}_j\texttt{(0)}$.

We use the Monotone3sat problem, mentioned in Theorem 2 above, to prove that $\mathrm{DP}_{Perf,Imp,Prop}(\,\Upsilon^{Add-Rules}\,)$ is NP-hard. Given any monotone 3CNF formula ϕ, with positive clauses P and with negative clauses N, let

$$\mathrm{T}_\phi \quad = \quad \left\{ \begin{array}{ll} \bar{\texttt{c}}_j \texttt{ :- not(u}_{j1}\texttt{), not(u}_{j2}\texttt{), not(u}_{j3}\texttt{).} & \text{for } c_j = \{u_{j1}, u_{j2}, u_{j3}\} \ \in \ P \\ \bar{\texttt{c}}_j \texttt{ :- u}_{j1}\texttt{, u}_{j2}\texttt{, u}_{j3}\texttt{.} & \text{for } c_j = \{\bar{u}_{j1}, \bar{u}_{j2}, \bar{u}_{j3}\} \ \in \ N \end{array} \right\}$$

and

$$S_\phi \quad = \quad \left\{ \ \langle \bar{\texttt{c}}_j;\ \texttt{No} \rangle \quad \text{for } c_j \in \phi \ \right\}$$

We need only show that there is a set of additions leading to a perfect theory iff ϕ has a satisfying assignment. Let $f : U \mapsto \{0,1\}$ be an assignment satisfying ϕ, and let T′ be a theory formed from T_ϕ by adding $\texttt{u}_i$ iff $f(u_i) = 1$. Notice T′ is perfect: For each $c_j = \{u_{j1}, u_{j2}, u_{j3}\} \in P$, T′ includes a $\texttt{u}_{ji}$, which means the associated $\texttt{not(u}_{ji}\texttt{)}$ fails, and so T′ will not entail $\bar{\texttt{c}}_j$. Similarly, for each $c_j = \{\bar{u}_{j1}, \bar{u}_{j2}, \bar{u}_{j3}\} \in N$, T′ does *not* entail some $\texttt{u}_{ji}$, which again means T′ will not entail $\bar{\texttt{c}}_j$. As no other addition is useful (in particular, adding $\bar{\texttt{c}}_j$ is counterproductive), finding a perfect T′ in $\Upsilon^{Add-Rules}(\mathrm{T}_\phi)$ means there is a satisfying assignment, formed by setting $f(u_i) = 1$ iff T′ includes $\texttt{u}_i$.

To deal with $\mathrm{DP}_{Perf,Pur,PC1}(\,\Upsilon^{Add-Rules}\,)$: Change Theorem 2's $\mathrm{T}_\varphi^{(PC)}$ theory by replacing each "$\texttt{u}_i\texttt{(0)}$." atomic clause with "$\texttt{u}_i\texttt{(0) :- notU}_i$.". Now notice the only additions, to the end of the theory, that can change the first answer returned to any $\texttt{c}_j\texttt{(X)}$ query will be atomic clauses of the form $\texttt{notU}_i$. This will cause $\texttt{u}_i\texttt{(0)}$ to be the (first) answer to the $\texttt{u}_i\texttt{(Z)}$ subquery, etc.

For $\text{MAX}_{Opt,Imp,Prop}(\Upsilon^{Add-Rules})$, we again use the reduction from MaxIndSet: Let

$$\mathrm{T}_{AR}^{(Prop)} = \left\{ \begin{array}{ll} \left.\begin{array}{l} \texttt{n}̄_j \texttt{ :- b.} \\ \bar{\texttt{n}}_j \texttt{ :- m}_j\texttt{.} \end{array}\right\} & \text{for } n_j \in N \\ \texttt{b :- not(m}_{i_1}\texttt{), not(m}_{i_2}\texttt{).} & \text{for } e_i = \langle n_{i_1}, n_{i_2} \rangle \in E \end{array} \right\}$$

and

$$S_{AR}^{(Prop)} = \{ \langle \bar{\texttt{n}}_i;\ \texttt{No} \rangle \quad \text{for } n_i \in N \}$$

Notice the accuracy of the initial $\mathrm{T}_{AR}^{(Prop)}$ is $\mathrm{A}(\mathrm{T}_{AR}^{(Prop)}) = 0$, as $\mathrm{T}_{AR}^{(Prop)}$ entails `b`, and therefore $\mathrm{T}_{AR}^{(Prop)}(\bar{\texttt{n}}_j) = \texttt{Yes}$. The only way to prevent this is by adding in some $\texttt{m}_i$ clauses — in fact, the revision system needs to add at least one of $\{\texttt{m}_{i_1}, \texttt{m}_{i_2}\}$ for each $e_i = \langle n_{i_1}, n_{i_2} \rangle \in E$. We can therefore view $\texttt{m}_i$ as meaning the node $n_i \in \texttt{N}$ is *not* selected in the independent set; and so $\texttt{not(m}_i\texttt{)}$ holds if the node n_i is included.

In general, let R be the set of $\texttt{m}_i$s that a revision process does *not* add in (which means the corresponding n_i is in the proposed independent set). By the arguments above, the resulting theory will have an accuracy score of $\frac{|R|}{|N|}$ if R corresponds to an independent set, and 0 otherwise. The rest of this proof follows the arguments used in Theorem 5.

(Note that it does not matter where the atomic clauses are added, for either $\text{DP}_{Perf,Imp,Prop}(\Upsilon^{Add-Rules})$ or $\text{MAX}_{Opt,Imp,Prop}(\Upsilon^{Add-Rules})$.)

To deal with $\text{MAX}_{Opt,Pur,PC1}(\Upsilon^{Add-Rules})$, use the theory

$$\mathrm{T}_{AR}^{(PC)} = \left\{ \begin{array}{ll} \left.\begin{array}{l} \texttt{n}_i\texttt{(1) :- m}_i\texttt{(0).} \\ \texttt{n}_i\texttt{(Z) :- b}_{|E|}\texttt{(Z).} \\ \texttt{m}_i\texttt{(0) :- xfer}_i\texttt{.} \\ \texttt{m}_i\texttt{(1).} \end{array}\right\} & \texttt{for } n_i \in N \\ \begin{array}{l} \texttt{b}_i\texttt{(Z) :- m}_{i_1}\texttt{(X}_a\texttt{), m}_{i_2}\texttt{(X}_b\texttt{), and2(X}_a\texttt{, X}_b\texttt{, Z}_i\texttt{),} \\ \qquad\quad \texttt{b}_{i-1}\texttt{(Z}_{i-1}\texttt{), or2(Z}_i\texttt{, Z}_{i-1}\texttt{, Z).} \end{array} & \texttt{for } \langle \texttt{n}_{i_1}, \texttt{n}_{i_2} \rangle \in E \\ \texttt{and2(1, 1, 1).} \quad \texttt{or2(1, 1, 1).} & \\ \texttt{and2(0, 1, 0).} \quad \texttt{or2(0, 1, 1).} & \\ \texttt{and2(1, 0, 0).} \quad \texttt{or2(1, 0, 1).} & \\ \texttt{and2(0, 0, 0).} \quad \texttt{or2(0, 0, 0).} & \end{array} \right\}$$

where the body of the $\texttt{b}_1\texttt{(Z)}$ clause only includes the first 3 literals:

$$\texttt{b}_1\texttt{(Z) :- m}_{1a}\texttt{(X}_a\texttt{), m}_{1b}\texttt{(X}_b\texttt{), and2(X}_a\texttt{, X}_b\texttt{, Z) .}$$

The queries here are

$$S_{AR}^{(PC)} = \{ \langle \texttt{n}_i\texttt{(X); Yes[X=0]} \rangle \quad \text{for } n_i \in N \}$$

As in the previous proof, the initial theory (here $\mathrm{T}_{AR}^{(PC)}$) has an accuracy score of 0, as $\texttt{m}_i\texttt{(0)}$ is not entailed and the first answer to each $\texttt{n}_i\texttt{(X)}$ is $\texttt{Yes[X=1]}$, as $\texttt{b}_{|E|}\texttt{(Z)}$ returns $\texttt{Yes[Z=1]}$ as each $\texttt{m}_i\texttt{(X)}$ returns $\texttt{Yes[X=1]}$. One way to prevent this is to change the theory so that some $\texttt{m}_i\texttt{(X)}$s instead return $\texttt{Yes[X=0]}$, which we can do by adding the corresponding

$\texttt{xfer}_i$ atomic clauses. Moreover, given the structure of the theory, and the fact that we can only add clauses to the end of the theory, this is actually the only approach. Notice we need to add at least one of $\{\texttt{xfer}_i, \texttt{xfer}_j\}$ for each $\langle n_i, n_j \rangle \in E$ (otherwise the first answer to $\texttt{b}_{|E|}(Z)$ will be $\texttt{Yes}[\texttt{Z}=\texttt{1}]$, leading to an accuracy of 0). The rest of this proof follows the proof above. □ (Theorem 9)

References

[1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. In *FOCS*, 1992.

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

[4] Mark Boddy and Thomas Dean. Solving time dependent planning problems. Technical report, Brown University, 1988.

[5] C. Boutilier. Revision sequences and nested conditionals. In *Proceedings of IJCAI-93*, pages 519–525, 1993.

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

[7] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. *Annals of Mathematical Statistics*, 23:493–507, 1952.

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

[9] William W. Cohen. PAC-learning recursive logic programs: Efficient algorithms. *Journal of Artificial Intelligence Research*, 2:500–539, 1995.

[10] William W. Cohen. PAC-learning recursive logic programs: Negative results. *Journal of Artificial Intelligence Research*, 2:541–573, 1995.

[11] William W. Cohen. PAC-learning non-recursive prolog clauses. *Artificial Intelligence*, 79(1):1–38, 1996.

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

[13] Mukesh Dalal. Investigations into a theory of knowledge base revision: Preliminary report. In *Proceedings of AAAI-88*, pages 475–479, 1988.

[14] A. Darwiche and J. Pearl. On the logic of iterated belief revision. In *TARK-94*, pages 5–23, 1994.

[15] Thomas G. Dietterich. Machine learning. *Annual Review of Computer Science*, 4:255–306, 1990.

[16] William F. Dowling and Jean H. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formula. *Journal of Logic Programming*, 3:267–84, 1984.

[17] 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.

[18] S. Dzeroski, S. Muggleton, and S. Russell. PAC-learnability of determinate logic programs. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, Pittsburgh, 1992.

[19] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revison, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[20] Rick Evertsz. The automated analysis of rule-based systems, based on their procedural semantics. In *Proceedings of IJCAI-91*, pages 22–27, 1991.

[21] M. Freund and D. Lehmann. Belief revision and rational inference. Technical Report TR-94-16, Hebrew University, 1994.

[22] N. Friedman and J. Halpern. Belief revision: A critique. In *KR-96*, 1996.

[23] Peter Gardenfors. *Knowledge in Flux: Modeling the Dynamics of the Epistemic States.* Bradford Book, MIT Press, Cambridge, MA, 1988.

[24] 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.

[25] G. Gogic, C. H. Papadimitriou, and M. Sideri. Incremental recompilation of knowledge. In *Proceedings of AAAI-94*, pages 922–927, 1994.

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

[27] Russell Greiner. The complexity of theory revision. In *Proceedings of IJCAI-95*, 1995.

[28] Russell Greiner. The complexity of theory revision. *Artificial Intelligence*, 1999. to appear;

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

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

[31] Hirofumi Katsuno and Alberto Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of KR-91*, pages 387–94, Boston, April 1991.

[32] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic leaning. In *Proceedings COLT-92*, pages 341–352. ACM Press, 1992.

[33] John E. Laird, Paul S. Rosenbloom, and Allan Newell. *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies.* Kluwer Academic Press, Hingham, MA, 1986.

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

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

[36] Charles X.F. Ling and Marco Valtorta. Some results on the computational complexity of refining certainty factors. *International Journal of Approximate Reasoning*, 5:121–148, 1991.

[37] Charles X.F. Ling and Marco Valtorta. Refinement of uncertain rule bases via reduction. *International Journal of Approximate Reasoning*, 13:95–126, 1995.

[38] S. Muggleton and W. Buntine. Machine invention of first order predicates by inverting resolution. In *Proceedings of IML-88*, pages 339–351. Morgan Kaufmann, 1988.

[39] S.H. Muggleton. *Inductive Logic Programming.* Academic Press, 1992.

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

[41] A. Carlisle Scott, Jan E. Clayton, and Elizabeth L. Gibson. *A Practical guide to knowledge acquisition.* Addison-Wesley Pub Co., Reading, MA, 1991.

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

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

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