Appears in the
Proceedings of the Third International Conference on Knowledge Representation and Reasoning,
25-29 October 1992, Cambridge, MA.

Learning Useful Horn Approximations

Russell Greiner[*]
755 College Road East
Siemens Corporate Research
Princeton, NJ 08540
greiner@learning.siemens.com

Dale Schuurmans
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
dale@cs.toronto.edu

Abstract

While the task of answering queries from an arbitrary propositional theory is intractable in general, it can typically be performed efficiently if the theory is Horn. This suggests that it may be more efficient to answer queries using a "Horn approximation"; *i.e.*, a horn theory that is semantically similar to the original theory. The utility of any such approximation depends on how often it produces answers to the queries that the system actually encounters; we therefore seek an approximation whose expected "coverage" is maximal. Unfortunately, there are several obstacles to achieving this goal in practice: *(i)* The optimal approximation depends on the query distribution, which is typically not known *a priori*; *(ii)* identifying the optimal approximation is intractable, even given the query distribution; and *(iii)* the optimal approximation might be too large to guarantee tractable inference. This paper presents an approach that overcomes (or side-steps) each of these obstacles. We define a learning process, AdComp, that uses observed queries to estimate the query distribution "online", and then uses these estimates to hill-climb, efficiently, in the space of size-bounded Horn approximations, until reaching one that is, with provably high probability, effectively at a local optimum.

1 Introduction

Many performance systems compute answers to queries based on the information present in a knowledge base. Unfortunately, this can involve reasoning from an arbitrary propositional theory, which is inherently intractable (assuming $P \neq NP$) [Coo71, GJ79]. We describe a technique that "approximates" an arbitrary theory, transforming it into a representation that admits more efficient, if less categorical, reasoning [EBBK89]. In particular, our work extends the "knowledge compilation" method of Selman and Kautz [SK91]: Given a general propositional theory Σ, their compiler will compute a pair of "bracketing" Horn theories S and W, with the property that $S \models \Sigma \models W$.[1] Figure 1 shows how the resulting "compiled system" $PS = PS^{\alpha}(S, W, \Sigma)$ uses these bracketing theories to determine whether a query σ follows from Σ. If $W \models \sigma$, PS returns "yes"; otherwise, if $S \not\models \sigma$, then PS returns "no". Notice that these are the correct answers; *i.e.*, $W \models \sigma$ guarantees that $\Sigma \models \sigma$, and $S \not\models \sigma$ guarantees that $\Sigma \not\models \sigma$. Moreover, these tests are efficient; in fact, linear in the size of S, W and σ [DG84], provided $\neg\sigma$ is Horn.[2]

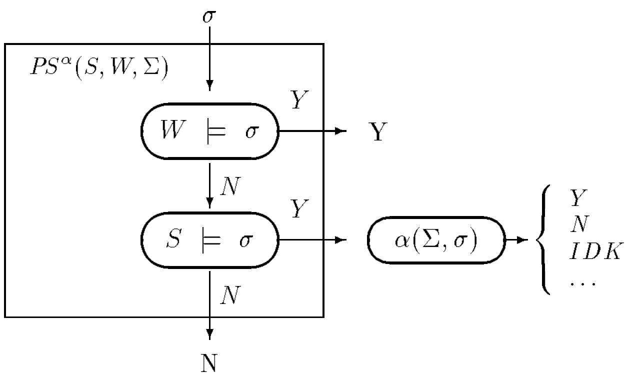

Figure 1: Flow Diagram of $PS^{\alpha}(S, W, \Sigma)$ addressing $\Sigma \models^{?} \sigma$

[*]Some of this work was performed at the University of Toronto, where it was supported by the Institute for Robotics and Intelligent Systems, and by an operating grant from the National Science and Engineering Research Council of Canada. Both authors thank Bart Selman, Alon Levy, Radford Neal, Sheila McIlraith, Narendra Gupta and the anonymous referees for providing many helpful comments on this paper.

[1]We call each such S a "Strengthening" of the initial theory, and each such W an "Weakening". We assume each general theory is in clausal form — *i.e.*, expressed as a set (conjunction) of clauses, where each clause is a set (disjunction) of atomic literals, each either positive or negative. A theory is Horn if each clause includes at most one positive literal.

[2]We can actually allow the query σ to be a conjunction of "Horn-dual" propositions (CHD), where a proposition σ is a Horn-dual iff its negation $\neg\sigma$ is horn. Notice CHD strictly includes CNF.

This paper extends [SK91]'s interesting results by addressing several unresolved issues.

ISSUE 1: WHICH α? It is not clear what *PS* should do if $W \not\models \sigma$ and $S \models \sigma$. For instance, we can consider various different classes of performance systems, each identified by its superscript: In particular, a PS^{IDK} system will simply return IDK (for "I don't know") in this situation, PS^{GUE} will "guess" at an answer, while the sound PS^{SND} will spend as long as necessary to compute whether $\Sigma \models \sigma$.

Of course, we want this problematic situation to occur rarely; we therefore prefer S and W theories that cover a maximal number of queries, as this means a minimal number of queries will fall through to the final α stage. [SK91] suggests restricting S (resp., W) to be a "weakest Strengthening", s_w (resp., a "strongest Weakening", w_s), which are the obvious extrema:

$$s_w(\Sigma, S) \iff \forall\, T\ [S \models T \models \Sigma \ \&\ \text{Horn}(T)] \ \Rightarrow\ S \equiv T$$

$$w_s(\Sigma, W) \iff \forall\, T\ [\Sigma \models T \models W \ \&\ \text{Horn}(T)] \ \Rightarrow\ W \equiv T$$

That article argues that such extrema are appropriate, as they cover a maximal number of queries. (To illustrate this idea, let W be a weakening that is not the strongest one — *i.e.*, $\Sigma \models w_s \models W$ where $W \not\models w_s$. Then there are queries σ such that $PS(S, w_s, \Sigma)$ would return Yes quickly, but $PS(S, W, \Sigma)$ will fall through to the problematic $\alpha(\Sigma, \sigma)$ step.)

There are, however, several complications associated with these extrema.

ISSUE 2: INTRACTABLE COMPILATION. The task of finding either extremum is intractable [SK91, p906], meaning they cannot be found efficiently (if $P \neq NP$).

ISSUE 3: MULTIPLE STRENGTHENINGS. There can be several weakest strengthenings. (For example, $\{a\}$ and $\{b\}$ each qualify as a weakest strengthening for $a \vee b$; *i.e.*, each satisfy $s_w(\{a \vee b\}, \cdot)$.)

ISSUE 4: EXPONENTIALLY LARGE WEAKENING. The cost of the first step of the $PS(S, W, \Sigma)$ process — *viz.*, determining whether $W \models^? \sigma$ — is linear in the size of W; unfortunately (the unique) strongest weakening w_s can be *exponential* in the size of the initial Σ. This means the resulting $PS^{\alpha}(S, w_s, \Sigma)$ system can still be intractable (even if we use a trivial $\alpha = IDK$, that simply returns IDK), as its first step can require exponential time.[3]

The rest of the paper presents an algorithm, ADCOMP, that addresses (and/or explicitly side-steps) each of these concerns. Section 2 describes this algorithm and shows how it deals with most of the issues; Section 3 then discusses several extensions to cope with the remaining points. The proof that ADCOMP works correctly appears in Appendix A.

[3]Notice that we encounter different problems when seeking optimal weakenings and strengthenings: There is a unique optimal weakening, but its size can be exponentially larger than $|\Sigma|$. By contrast, there can be many different optimal strengthenings; however each is essentially the same size as Σ; Subsection 2.4.

2 The ADCOMP Algorithm

The basic idea underlying our approach is to learn a reasonably-sized approximation that is likely to be good enough for the anticipated queries. Subsection 2.1 first motivates this approach; the rest of the section describes the ADCOMP algorithm ("ADaptive COMPiler") that implements these ideas. Subsection 2.2 states the fundamental theorem that specifies ADCOMP's functionality (whose proof appears in Appendix A). Subsection 2.3 provides the statistical foundations to motivate why this algorithm is feasible. Subsections 2.4 and 2.5 then present further details of the structure of the ADCOMP algorithm; and Subsection 2.6 discusses the algorithm's computational efficiency.

2.1 Our Approach

Tractable Inference. Given our objective of finding a representation of the given theory that admits efficient reasoning, we will (for now) consider only *polynomial-sized weakenings* (as this guarantees that $W \models \sigma$ can be answered efficiently) and to $PS^{IDK}(S, W, \Sigma)$ systems (as they are guaranteed to run efficiently, simply terminating with IDK whenever $W \not\models \sigma$ and $S \models \sigma$). These restrictions avoid the problems mentioned ISSUE 1 (WHICH α?) and ISSUE 4 (EXPONENTIALLY LARGE WEAKENING); Extension 6 in Section 3 will later return to these issues.

To state this more precisely: Given any propositional theory Σ, define $\text{Approx}_K(\Sigma)$ to be to the set of all Horn approximations of Σ whose sizes are at most K, *i.e.*,

$$\langle S, W\rangle \in \text{Approx}_K(\Sigma) \iff S \models \Sigma \models W \quad \& \quad \text{Horn}(W) \quad \& \quad \text{Horn}(S) \quad \& \quad |S| \leq K \quad \& \quad |W| \leq K$$

where the size of a horn theory $|T|$ is the number of clauses in T.[4] We identify each such Horn approximation $\langle S, W\rangle \in \text{Approx}_K(\Sigma)$ with the associated performance system $PS^{IDK}(S, W, \Sigma)$.

Utility of Horn Approximations. ISSUE 3 (MULTIPLE STRENGTHENINGS) noted there are many possible weakest strengthenings of a given theory; there are also many different K-sized strongest weakenings. How can we decide which to use? We adopt a pragmatic position: the optimal system is the one that has the *best expected performance* over the natural distribution of queries, based on a scoring function. For now, we define the scoring function to be simply the approximation's *coverage*.[5] Given any approximation $\langle S, W\rangle$ and query σ, let $c(\langle S, W\rangle, \sigma) \stackrel{def}{=} d(W, \sigma) + (1 - d(S, \sigma))$ where

$$d(T, \sigma) \stackrel{def}{=} \begin{cases} 1 & \text{if } T \models \sigma \\ 0 & \text{otherwise .} \end{cases}$$

[4]As the number of literals in each clause is at most L (where L = total number of variables), this measure is within a constant factor of the other obvious ways of measuring the size of a theory.

[5]Extension 6 in Subsection 3 considers other scoring functions. Notice higher scores are preferable.

Hence, $c(\langle S, W\rangle, \sigma) = 1$ iff σ is "covered" by $\langle S, W\rangle$, in that either $W \models \sigma$ or $S \not\models \sigma$.

This cost function evaluates $\langle S, W\rangle$'s performance for a single query. Our approximations, however, will have to solve an entire ensemble of problems; we clearly prefer the approximation that is best overall. We therefore define the "utility of the approximation $\langle S, W\rangle$" to be the expected value of this $c(\langle S, W\rangle, \cdot)$ scoring function, over the natural distribution of queries.

To state this more precisely: Given $\mathcal{Q}$, the set of all possible (CHD) queries, let $P\colon \mathcal{Q} \mapsto [0, 1]$ be the stationary distribution of the queries, where $P(q)$ is the probability of encountering the query $q \in \mathcal{Q}$. Then the utility measure used to evaluate an approximation $\Upsilon = \langle S, W\rangle$ is its expected score with respect to P,

$$\mathrm{C}[\,\Upsilon\,] \quad \stackrel{def}{=} \quad E[\,c(\Upsilon, \boldsymbol{\sigma})\,] \;=\; \sum_{\sigma\in\mathcal{Q}} P(\sigma) \times c(\Upsilon, \sigma)\ .$$

Our basic goal is to identify an "optimal K-sized approximation", which is an approximation $\Upsilon_{opt} \in \mathrm{Approx}_K(\Sigma)$ whose expected score is maximal:

$$\begin{aligned} &\Upsilon_{opt} \in \mathrm{Approx}_K(\Sigma) \quad \& \\ &\forall \Upsilon \in \mathrm{Approx}_K(\Sigma) \;\; \mathrm{C}[\,\Upsilon_{opt}\,] \;\geq\; \mathrm{C}[\,\Upsilon\,] \end{aligned}$$

Hence, we are seeking a (reasonably-sized) horn approximation that is good *for the given distribution of queries.* Unfortunately, there are two major obstacles to achieving this goal; these are described in the next two points, and then addressed in the next subsections.

Learning. First, under the realistic assumption that the distribution P is unknown, there is no *a priori* way of determining the values of $\mathrm{C}[\,\cdot\,]$, and hence of determining which Υ is optimal. Fortunately, we can use *learning techniques* (read "statistical methods") to reliably estimate this distribution, and then use these estimates to compute a near-optimal approximation; see Subsection 2.3.

Hill-Climbing. Second, even given this distribution P, the task of finding an optimal approximation is intractable; this is the essence of ISSUE 2 (INTRACTABLE COMPILATION). The ADCOMP process, defined below, avoids this problem by *hill-climbing* in the space of horn approximations, climbing from some initial approximation to successively better ones, until reaching a local peak.

2.2 ADCOMP's Behavior

The basic code for the ADCOMP algorithm appears in Figure 2. Its inputs are an initial theory Σ, error and confidence parameters $\epsilon, \delta > 0$, and a resource bound, the polynomial function $K(\cdot)$. Its output is a near-optimal approximation $\langle S_n, W_m\rangle$, as specified below. ADCOMP observes a sequence of queries,[6] printing out an answer (`Yes`, `No` or `IDK`) to each, as it computes its approximations of Σ.

ADCOMP makes use of a particular set of transformations, $\mathcal{T}^S \cup \mathcal{T}^W$, each mapping approximations to approximations. Subsections 2.4 and 2.5 define these transformations more precisely; for now just observe that each $\tau^S \in \mathcal{T}^S$ maps strengthenings to strengthenings and the set $\mathrm{NEIGHS}[S] = \{\tau^S(S) \mid \tau^S \in \mathcal{T}^S\}$ defines S's neighbors. Similarly, each $\tau^W \in \mathcal{T}^W$ maps weakenings to weakenings, and W's neighbors are $\mathrm{NEIGHW}(W) = \{\,\tau^W(W) \mid \tau^W \in \mathcal{T}^W\,\}$.

In essence, ADCOMP first computes an initial S_1 and then climbs from S_1 to one of its neighbors, $S_2 \in \mathrm{NEIGHS}[S_1]$, if S_2 is statistically likely to be superior to S_1, based on the sequence of observed queries. This constitutes one hill-climbing step; in general, ADCOMP will perform many such steps, climbing from S_1 to S_2 to S_3, etc., until reaching a near optimal S_n. In parallel with this process, ADCOMP also uses these queries to hill-climb from an initial (computed) W_1 to a neighbor $W_2 \in \mathrm{NEIGHW}[W_1]$, and then on to $W_3 \in \mathrm{NEIGHW}[W_2]$, etc., until reaching a near optimal W_m. ADCOMP returns the resulting $\langle S_n, W_m\rangle$, whose expected score is, with probability at least $1-\delta$, at an "ϵ-local optimum" with respect to these transformations $\mathcal{T}^S \cup \mathcal{T}^W$:

Theorem 1 *The* ADCOMP$(\Sigma, \epsilon, \delta, K(\cdot))$ *process incrementally produces a series of weakenings $\langle W_1, W_2, \ldots, W_m\rangle$ and (independently) a sequence of strengthenings $\langle S_1, S_2, \ldots, S_n\rangle$ such that, with probability at least $1-\delta$,*

1. *each successive approximation has an expected score that is strictly better than its predecessor's; i.e.,*
$$\begin{aligned} \mathrm{C}[\,\langle S_{i+1}, W_j\rangle\,] \;&>\; \mathrm{C}[\,\langle S_i, W_j\rangle\,] \\ \mathrm{C}[\,\langle S_i, W_{j+1}\rangle\,] \;&>\; \mathrm{C}[\,\langle S_i, W_j\rangle\,] \end{aligned}$$
and
2. *the final approximation $\langle S_n, W_m\rangle$ is an ϵ-local optimum; i.e., its expected score is within ϵ of the best expected score among its neighbors:*
$$\begin{aligned} \forall\, \tau \in \mathcal{T}^S\colon\;\; &\mathrm{C}[\,\langle S_n, W_m\rangle\,] \;\geq\; \mathrm{C}[\,\langle \tau(S_n), W_m\rangle\,] - \epsilon \\ \forall\, \tau \in \mathcal{T}^W\colon\;\; &\mathrm{C}[\,\langle S_n, W_m\rangle\,] \;\geq\; \mathrm{C}[\,\langle S_n, \tau(W_m)\rangle\,] - \epsilon. \end{aligned}$$

Moreover, ADCOMP *requires only polynomial time (and hence only a polynomial number of samples) to decide whether to move from S_i to S_{i+1} (resp., from W_j to W_{j+1}) or terminate with a final S_n (resp., with a final W_m).* □.

(The proof appears in Appendix A.)

2.3 Statistical Foundations

Notice first that $\mathrm{C}[\,\langle S_i, W_j\rangle\,] = \mathrm{D}[\,W_j\,] + (1 - \mathrm{D}[\,S_i\,])$, where

$$\mathrm{D}[\,T\,] \;=\; E[\,d(T, \boldsymbol{\sigma})\,] \;=\; \sum_{\sigma\in\mathcal{Q}} P(\sigma) \times d(T, \sigma)$$

is the likelihood that the theory T will entail a query, over the distribution of queries. As we are only considering transformations that affect only one of W_j or S_i, an approximation $\langle S_i, W_j\rangle$ is, with probability at least

[6] These queries are from the user of the performance system, who is posing queries relevant to one of his application. In general, we need only assume that he is drawing these queries from a stationary distribution, and that this is the same distribution that will be used later, when the resulting performance system $PS^\alpha(S_n, W_m, \Sigma)$ is actually being used.

Algorithm AdComp(Σ, δ, ϵ, $K(\cdot)$)

Init: $j \leftarrow 0$, $\mathtt{S}_1 \leftarrow \textsc{InitialS}(\Sigma_H, \Sigma_N)$, $\langle \mathtt{W}_1, \Omega_1 \rangle \leftarrow \textsc{InitialWN}(\Sigma_H, \Sigma_N, K(\Sigma))$,
`FoundGoodS` $\leftarrow$ `False`, `FoundGoodW` $\leftarrow$ `False`

Loop

- $j \leftarrow j+1$, $\textsc{NeighS} \leftarrow \{\tau_k^S(\mathtt{S}_j)\}_k$, $\textsc{NeighW} \leftarrow \{\tau_k^W[\mathtt{W}_j, \Omega_j](\mathtt{W}_j)\}_k$

$$n_j \quad \leftarrow \quad \left\lceil \frac{2}{\epsilon^2} \ln \frac{2\, j^2\, \pi^2\, \max\{|\textsc{NeighS}|,\, |\textsc{NeighW}|\}}{3\,\delta} \right\rceil \tag{1}$$

`/* Get Samples, Print Answers */`

Get n_j `samples,` $\mathtt{Q}_j = \{\sigma_1, \sigma_2, \ldots, \sigma_{n_j}\}$ `from the user`

For `each sample` $\sigma_i \in \mathtt{Q}_j$ `do`

  If `for some` $\mathtt{W}' \in \{\mathtt{W}_j\} \cup \textsc{NeighW}$, $\mathtt{W}' \models \sigma_i$

   then `Print` "σ_i : `Yes`"

  ElseIf `for some` $\mathtt{S}' \in \{\mathtt{S}_j\} \cup \textsc{NeighS}$, $\mathtt{S}' \not\models \sigma_i$

   then `Print` "σ_i : `No`"

  Else `Print` "σ_i : `IDK`"

End For

`/* Iterate or Terminate, wrt Strengthenings */`

If $\neg$`FoundGoodS` **then**

  If `for some` $\mathtt{S}' \in \textsc{NeighS}$,

$$d(\mathtt{S}_j, \mathtt{Q}_j) - d(\mathtt{S}', \mathtt{Q}_j) \quad \geq \quad \frac{\epsilon}{2} \tag{2}$$

  then $\mathtt{S}_{j+1} \leftarrow \mathtt{S}'$,

  Else `/* Here,` $d(\mathtt{S}_j, \mathtt{Q}_j) - d(\mathtt{S}', \mathtt{Q}_j) < \frac{\epsilon}{2}$ `for all` $\mathtt{S}' \in \textsc{NeighS}$ `*/`

   `FoundGoodS` $\leftarrow$ `True`

   $\mathtt{S}_{final} \leftarrow \mathtt{S}_j$

End If

`/* Iterate or Terminate, wrt Weakenings */`

If $\neg$`FoundGoodW` **then**

  If `for some` $\mathtt{W}' \in \textsc{NeighW}$,

$$d(\mathtt{W}', \mathtt{Q}_j) - d(\mathtt{W}_j, \mathtt{Q}_j) \quad \geq \quad \frac{\epsilon}{2} \tag{3}$$

  then $\mathtt{W}_{j+1} \leftarrow \mathtt{W}'$, $\Omega_{j+1} \leftarrow \textsc{UpdateN}(\mathtt{W}_j, \Omega_j)$

  Else `/* Here,` $d(\mathtt{W}', \mathtt{Q}_j) - d(\mathtt{W}_j, \mathtt{Q}_j) < \frac{\epsilon}{2}$ `for all` $\mathtt{W}' \in \textsc{NeighW}$ `*/`

   `FoundGoodW` $\leftarrow$ `True`

   $\mathtt{W}_{final} \leftarrow \mathtt{W}_j$

End If

Until: `FoundGoodS & FoundGoodW`

`Return` $PS^{IDK}(\mathtt{S}_{final},\ \mathtt{W}_{final},\ \Sigma)$

End AdComp

Figure 2: Basic AdComp algorithm (see description in Subsection 2.2)

$1-\delta$, within ϵ of a local optimum if W_j is within ϵ of a locally optimal weakening, and S_i is within ϵ of a locally optimal strengthening, each with probability at least $1-\frac{\delta}{2}$. We can therefore decouple the task of finding a good strengthening from that of finding a good weakening, and handle each separately.

We would like AdComp to climb from a current S_j to a new $S_{j+1} \in \text{NeighS}[S_j]$ if S_{j+1} is statistically likely to be strictly better than S_j. (Similarly, from W_i to $W_{i+1} \in \text{NeighW}[W_i]$, etc.) The next subsections define appropriate sets of transformations, $\mathcal{T}^S$ and $\mathcal{T}^W$; the rest of this subsection specifies when to make each such transition.

When is S_α better than S_β? By definition, S_α is better than S_β whenever $\text{D}[S_\alpha] < \text{D}[S_\beta]$, or equivalently, when $\text{D}[S_\beta] - \text{D}[S_\alpha] > 0$. The value of $\text{D}[S_\beta] - \text{D}[S_\alpha]$ depends on the distribution, P, which unfortunately is unknown.

We can however use a set of samples to estimate this quantity, and then use statistical methods to bound our confidence of the accuracy of these estimates. To do this, let the variable $\Delta_i = d(S_\beta, \sigma_i) - d(S_\alpha, \sigma_i)$ be the difference in the coverage between S_β and S_α, for the query σ_i. As each query is selected according to a fixed distribution, these Δ_is are independent, identically distributed random variables whose common mean is $\mu = \text{D}[S_\beta] - \text{D}[S_\alpha]$, which is the quantity we want to estimate. Now let

$$\begin{aligned} Y_n &= \frac{1}{n}\sum_{i=1}^{n} d(S_\beta, \sigma_i) - d(S_\alpha, \sigma_i) \\ &= d(S_\beta, \{\sigma_i\}_{i=1}^n) - d(S_\alpha, \{\sigma_i\}_{i=1}^n) \end{aligned}$$

be the sample mean of n samples.[7] This average will tend to the population mean μ as $n \to \infty$; *i.e.*, $\mu = \lim_{n \rightsquigarrow \infty} Y_n$. Chernoff bounds [Che52] provide the probable rate of convergence: the probability that "Y_n is more than $\mu + \beta$" goes to 0 exponentially fast as n increases; and, for a fixed n, exponentially as β increases. Formally,[8]

$$\begin{aligned} Pr[Y_n > \mu + \beta] &\leq e^{-2n\beta^2} \\ Pr[Y_n > \mu - \beta] &\leq e^{-2n\beta^2} \end{aligned} \tag{4}$$

The AdComp algorithm uses these formulae and the observed values of various $d(S_j, \sigma)$ and $d(\tau_k^S(S_j), \sigma)$, to determine both how confident we should be that $\text{D}[S_j] > \text{D}[S']$ and also whether any "$\mathcal{T}^S$-neighbor" of S_j (*i.e.*, any $\tau_k^S(S_j)$) is more than ϵ better than S_j. (Of course, similar conditions apply for strengthenings: W_α is better than W_β whenever $\text{D}[W_\alpha] - \text{D}[W_\beta] > 0$, etc.) See the proof in Appendix A.

2.4 Finding a good Horn Strengthening

A "horn-strengthening" of the clause $\gamma = \{a_1, \ldots, a_k, \neg b_1, \ldots, \neg b_\ell\}$ is any maximal clause that is a subset of γ and is Horn — *i.e.*, each horn-strengthening is formed by simply discarding all but one of the positive literals. Here, there are k horn strengthenings of this γ, each of the form $\gamma_j = \{a_j, \neg b_1, \ldots, \neg b_\ell\}$. (*E.g.*, the 2 horn strengthening of the non-Horn clause $\gamma \equiv a \vee b \vee \neg c \vee \neg d$ are $\gamma_1 \equiv a \vee \neg c \vee \neg d$ and $\gamma_2 \equiv b \vee \neg c \vee \neg d$.)

We can write $\Sigma = \Sigma_H \cup \Sigma_N$, where each element of Σ_H is a Horn clause, and each element of $\Sigma_N = \{\gamma^i\}_{i=1}^m$ is a non-Horn clause. [SK91] proves that each weakest strengthening is of the form $S_o = \Sigma_H \cup \Sigma_N'$, where $\Sigma_N' = \{\gamma^{i'}\}_{i=1}^m$ such that each $\gamma^{i'} \in \Sigma_N'$ is a horn-strengthening of some $\gamma^i \in \Sigma_N$. By identifying each Horn-strengthened theory with the "index" of the positive literal used (*i.e.*, ${\gamma^i}_j = \{a_j^i, \neg b_1^i, \ldots, \neg b_{\ell(i)}^i\}$), we can consider any Horn-strengthened theory to be a set of the form $S_{\langle j(1), j(2), \ldots, j(m)\rangle} = \Sigma_H \cup \{\gamma^1_{j(1)}, \gamma^2_{j(2)}, \ldots, \gamma^m_{j(m)}\}$. Notice that each of these strengthenings S_i is "small", in fact, $|S_i| = |\Sigma|$.

We can navigate about this space of Horn-strengthened theories by incrementing or decrementing the index of a specific non-Horn clause: That is, define the set of $2m$ transformations $\mathcal{T}^S = \{\tau_k^+, \tau_k^-\}_{k=1}^m$ where each τ_k^+ (resp., τ_k^-) is a function that maps one strengthening to another, by incrementing (resp., decrementing) the "index" of k^{th} clause — *e.g.*, $\tau_k^+(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,i_k+1,\ldots,5\rangle}$, and $\tau_i^-(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,i_k-1,\ldots,5\rangle}$. (Of course, the addition and subtraction operations wrap around.)

The AdComp process, therefore, starts with an arbitrary horn-strengthening — here, the $S_{\langle 1,1,\ldots,1\rangle}$ returned by $\text{InitialS}(\Sigma_H, \Sigma_N)$ — and then hill-climbs in this space of horn-strengthened theories, using the set of $\mathcal{T}^S$ transformations defined above. It will terminate on reaching an S_j which is an ϵ-local optimum. (Notice this S_j is not necessarily a weakest strengthening.)

2.5 Finding a good Horn Weakening

[SK91] proves that there is a unique optimal weakening, w_s, and presents the LUB algorithm for computing it. Their algorithm is equivalent to $\text{InitialWN}(\Sigma_H, \Sigma_N, \infty)$, using the process shown in Figure 3. The final $w_s = \text{InitialWN}(\Sigma_H, \Sigma_N, \infty)$ is the set of all horn implicates of the initial theory. It is easy to see that this w_s will have the largest possible $\text{D}[\cdot]$ value over all weakenings, for any distribution. Unfortunately, it can also be exponentially larger than the original theory [KS92].

As mentioned above, we avoid this potential blow-up by considering only weakenings of size at most $K = K(\Sigma)$, where $K(\cdot)$ is a user-supplied polynomial function.[9] Our goal, therefore, is to find the weakening of this size that is maximally categorical, over the distribution of queries.

AdComp performs a (tractable) hill-climbing search through the space of K-sized Horn weakenings of Σ, attempting to find one that has good empirical coverage (an ϵ-locally optimal expected score). As we are

[7] Notice $d(T, Q) = \frac{1}{|Q|}\sum_{\sigma \in Q} d(T, \sigma)$, for any theory T and any set of queries Q.

[8] See [Bol85, p. 12]. *N.b.*, these inequalities holds for essentially *arbitrary distributions*, not just normal distributions, subject only to the minor constraint that the random variables $\{d_i\}$ be bounded.

[9] To avoid degeneracies, we will assume that $K(\Sigma) \geq |\Sigma|$.

```
Algorithm InitialWN( Σ_H, Σ_N, K)
  W ← Σ_H;   Ω ← Σ_N;     Done ← True;     j ← 0
  Repeat
    j ← j + 1
    For each w ∈ W, and each n ∈ Ω
      If w and n resolve
      Then Let λ be the resolvent of w and n
           If    /* λ is NOT subsumed by any clause in W ∪ Ω */
                 ∀α ∈ W ∪ Ω :  α ⊈ λ
           Then Done ← False
                    /* Remove from W, Ω all clauses that λ subsumes */
                 W  ←  { w ∈ W | λ ⊈ w }
                 Ω  ←  { n ∈ Ω | λ ⊈ n }
                 If λ is horn,
                    Then /* add λ to W */
                          W   ←   W ∪ {λ}
                    Else  /* λ is non horn; add λ to Ω */
                          Ω   ←   Ω ∪ {λ}
                 End If
           End If
      End If
    End For
  Until  Done   or  |W| = K   or  |Ω| = K   or  j = K
  Return ⟨W, Ω⟩
End InitialWN
```

Figure 3: InitialWN Algorithm, adapted from lub in [SK91, p907]

only considering reasonably-sized theories, the result of this search is a useful Horn weakening of Σ from which we can perform *tractable* inference, thus addressing Issue 4 (Exponentially Large Weakening).

AdComp uses the InitialWN algorithm to generate an initial bounded weakening $\langle W_1, \Omega_1 \rangle = \text{InitialWN}(\Sigma_H, \Sigma_N, K)$. (Notice this process is efficient, as InitialWN will perform at most K iterations.) AdComp then uses a "1-step variant" of InitialWN to climb to successive weakenings. In particular, given $\langle W_j, \Omega_j \rangle$ at iteration j, AdComp will consider climbing from W_j using the transformations[10] $\mathcal{T}^W[W_j, \Omega_j] = \{\, \tau_{h_1,n_1,h_2} \,\}_{h_1,h_2 \in W_j;\, n_1 \in \Omega_j}$, where $\tau_{h_1,n_1,h_2}(W_j)$ returns

- $\{\}$, if h_1 does not resolve with n_1.
  Otherwise, let λ be the result of resolving h_1 with n_1.
- $\{\}$, if λ is not horn, or if λ is subsumed by any element of W_j.
  Otherwise,
- $W_j \cup \{\lambda\} - \{\eta_k\}_k$, if λ is horn and subsumes each $\eta_k \in W_j$. (Of course, there must be at least one such η.)
  Otherwise,
- $W_j \cup \{\lambda\}$, if $|W_j| < K$.
  Otherwise,
- $W_j \cup \{\lambda\} - \{h_2\}$, if λ is horn and does not subsume any clause in W_j (*i.e.*, τ_{h_1,n_1,h_2} replaces h_2 with λ in W_j). □.

[10]We write the set of transformations as $\mathcal{T}^W[W_j, \Omega_j]$ to indicate that it depends on the current weakening and its non-horn complement, $\langle W_j, \Omega_j \rangle$ and so can change from one weakening W_j to the next, W_{j+1}.

The resulting set of weakenings

$$\text{NeighW}(W_j) \;=\; \left\{ \; W \;\middle|\; \begin{array}{l} W = \tau_{h_1,n_1,h_2}(W_j)\ \& \\ \tau_{h_1,n_1,h_2} \in \mathcal{T}^W[W_j, \Omega_j] \\ \&\ W \neq \{\} \end{array} \right\}$$

includes all and only the non-$\{\}$ values $\tau_{h_1,n_1,h_2}(W_j)$.

Example: Imagine InitialWN returned the initial pair

$$\begin{array}{lcl} W_1 & = & \{\, \neg a,\ \neg b,\ d \,\} \\ \Omega_1 & = & \{\, a \vee b \vee \neg c,\ a \vee c \vee \neg d \,\} \end{array}$$

and let $K = 3$, meaning W_1 is filled to its capacity. Here, there are $|W_1| \times |\Omega_1| \times |W_1| = 3 \times 2 \times 3 = 18$ different transformations:

$$\mathcal{T}^W[W_1, \Omega_1] \;=\; \left\{ \begin{array}{lll} \tau_{\neg a,\, a\vee b\vee\neg c,\, \neg a} & \tau_{\neg a,\, a\vee b\vee\neg c,\, \neg b} & \tau_{\neg a,\, a\vee b\vee\neg c,\, d} \\ \tau_{\neg a,\, a\vee c\vee\neg d,\, \neg a} & \tau_{\neg a,\, a\vee c\vee\neg d,\, \neg b} & \tau_{\neg a,\, a\vee b\vee\neg d,\, d} \\ \tau_{\neg b,\, a\vee c\vee\neg d,\, \neg a} & \cdots & \\ & \cdots & \tau_{d,\, a\vee c\vee\neg d,\, d} \end{array} \right\}$$

Notice most transformations are degenerate, simply returning $\{\}$ — including all 3 of the form $\tau_{d,\, a\vee b\vee\neg c,\, \cdot}(W_1) = \{\}$ as d does not resolve with $a \vee b \vee \neg c$. The three transformations $\tau_{d,\, a\vee c\vee\neg d,\, \cdot}$ are also degenerate, as the resolvent of d and $a \vee c \vee \neg d$, namely $a \vee c$, is not horn. (But see Issue 5 in Subsection 3.)

To illustrate a non-degenerate transformation, observe $\tau_{\neg a,\, a\vee b\vee\neg c,\, d}(W_1) = \{b \vee \neg c,\ \neg a,\ \neg b\}$. Here, as $|W_1| = K$, we had to remove one element of W_1, namely d, to make space for $b\vee\neg c$, the resolvent of $\neg a$ and $a\vee b\vee\neg c$. If K had been larger, then $\tau_{\neg a,\, a\vee b\vee\neg c,\, d}(W_1)$ could simply add in this new $b \vee \neg c$, producing the 4-*element* weakening $\{b \vee \neg c,\ \neg a,\ \neg b,\ d\}$. □

If one of the $W' = \tau_{h_1,n_1,h_2}(W_j) \in$ NEIGHW(W_j) neighbors passes Equation 3 and becomes the new "current weakening" W_{j+1}, ADCOMP will use the UPDATEN process to compute a new Ω_{j+1}. UPDATEN first resolves each clause in the new W_{j+1} with each clause in Ω_j, then forms Ω_{j+1} by adding all the unsubsumed *non-Horn* clauses to Ω_j, and removing all subsumed clauses. (This is like the set of $\mathcal{T}^W[W_j, \Omega_j] = \{\tau_{h_1,n_1,h_2}\}$ transformations, but adding non-horn clauses to Ω_j, rather than horn clauses to W_j.) To keep $|\Omega_{j+1}| \leq K$, UPDATEN may have to delete an existing clause from Ω_j before adding a new resolvent. (This choice is arbitrary; we could, for example, simply remove the "oldest" clauses in Ω_j until the size bound is reached. Of course, there are many other approaches.)

Example: To continue with the earlier example, suppose $\tau_{\neg a,\, a\vee b\vee\neg c,\, d}(W_1) = \{b \vee \neg c,\ \neg a,\ \neg b\}$ passes Equation 3 and so becomes W_2. To compute Ω_2, UPDATEN first resolves each $h \in W_2$ with each $n \in \Omega_1$ (producing $\{\ b \vee \neg c,\ a \vee \neg c,\ a \vee b \vee \neg d,\ c \vee \neg d\ \}$), then adds to Ω_1 the non-horn propositions, forming $\{\ a \vee b \vee \neg c,\ a \vee c \vee \neg d,\ a \vee b \vee \neg d\ \}$. It then removes all subsumed clauses, leaving $\Omega_2 = \{\ a \vee c \vee \neg d,\ a \vee b \vee \neg d\ \}$. □

Notice each resulting theory W' can have no more clauses than W_j; hence, $|W'| \leq |W_j| \leq K$. Moreover, there are only $O(K^3)$ possible transformations and each of these new weakenings can be computed efficiently.

2.6 Efficiency

Each of ADCOMP's individual steps is tractable: The only potentially problematic steps involve asking whether $T \models^? \sigma$, where T is S_j, $\tau^S(S_j)$, W_j or $\tau^W(W_j)$. However, as each of these theories is horn and of bounded size (at most K), each of these computations is efficient.

As an important aside, notice that ADCOMP is using this battery of efficient tests to approximate the intractable $\Sigma \models^? \sigma$ test: correctly concluding that $\Sigma \models \sigma$ whenever either $W_j \models \sigma$ or $\tau^W(W_j) \models \sigma$ for any $\tau^W \in \mathcal{T}^W$, and that $\Sigma \not\models \sigma$ whenever $S_i \not\models \sigma$ or $\tau^S(S_i) \not\models \sigma$ for any $\tau^S \in \mathcal{T}^S$. ADCOMP will return IDK only if *none* of these tests succeeds — *i.e.*, if $W_j \not\models \sigma$ and $\tau^W(W_j) \not\models \sigma$ for all $\tau^W \in \mathcal{T}^W$, $S_i \models \sigma$ and $\tau^S(S_i) \models \sigma$ for all $\tau^S \in \mathcal{T}^S$.

ADCOMP can perform at most $|\mathcal{T}^W[W_j, \Omega_j]| + |\mathcal{T}^S|$ such derivations for each sample query, which is also a polynomial in the relevant parameters. Observe, moreover, that each iteration of the ADCOMP process can involve only a polynomial number of samples (n_j from Equation 1). The only part of this process that is not necessarily bounded by a polynomial is the number of iterations required. However, this is not necessarily problematic, as ADCOMP is essentially an anytime system [DB88], returning successively better and better horn approximations.

In fact, our ADCOMP can be viewed as a natural extension of the anytime compiler discussed in [SK91], as each system runs in parallel with a performance system that is using the current best approximation to return responses to the queries presented. ADCOMP differs (1) by using the set of observed queries to *guarantee* with provably high probability that each of the approximations truly is an improvement over its predecessors; (2) by avoiding the intractable $\Sigma \models \sigma$ test while learning; and (3) by guaranteeing that the approximations produced will admit tractable inference.

3 Extensions

This section discusses various extensions to our basic approach and the ADCOMP algorithm shown in Figure 2.

Extension 1. Minor Adjustments: ADCOMP uses a single value of K to bound the sizes of W_j and Ω_j and as the time limit for the INITIALWN process. An obvious variant would permit the user to supply several values, to separately specify the different size constraints and time bound.

Notice also that ADCOMP could perform a quick post-processing on the final strengthening S_n (resp., final weakening W_m), to convert this horn theory into a possibly smaller horn theory by resolving its clauses together and removing all subsumed expressions.

Extension 2. ADCOMP works in "batch" mode — using a collection of n_j (Equation 1) samples to decide whether to iterate from S_j to S_{j+1} or to stop improving the strengthening, and also whether to iterate from W_j to W_{j+1}, etc. [GJ92] presents PALO, a related algorithm (but designed for a different task) that can make these decisions after each individual sample. PALO can potentially require fewer samples on each iteration than ADCOMP, as PALO will consider climbing to a (probabilistically) better element or terminating, after seeing each sample. We have designed an algorithm, ADCOMP*, that basically uses PALO's techniques, but handles ADCOMP's application, and confirmed that ADCOMP* does satisfy Theorem 1.[11]

Extension 3. In general, ADCOMP must compute the values of $d(W', \sigma) - d(W_j, \sigma)$ for each W' that is a $\mathcal{T}^W$-neighbor of W_j. We can always obtain this information by constructing these neighboring W's, and using them to compute the relevant values of $d(W', \sigma)$. Alternatively, there are often ways of computing these values, based only on running the original W_j. As an example, imagine that $W_j \models \sigma$, and let $\{h_i\} \subseteq W_j$ be σ's support in W_j (*i.e.*, $\{h_i\} \models \sigma$). Clearly $W' \models \sigma$ will hold for each $W' \in$ NEIGHW(W_j) whenever $\{h_i\} \subseteq W'$ as well. Hence, we can guarantee that $d(W_j, \sigma) - d(W', \sigma) = 0$

[11] We chose to present the simpler ADCOMP version for pedagogic reasons, as ADCOMP* is much more difficult to explain.

in this context. (Of course, this same idea also applies to computing the values of $d(S_i,\sigma) - d(S',\sigma)$.)

Extension 4. We can consider using other transformations, especially when seeking an optimal weakening. For example, [KS92] suggests a way of shrinking the size of some horn weakening by adding new vocabulary terms to the initial theory; it would be easy to also include transformations that implement this idea. Other recent papers, including [DE92], propose other techniques for finding good (not necessarily horn) approximations.

Extension 5. The set of $\mathcal{T}^W$ transformations described in Subsection 2.5 will not always allow AdComp to explore the entire space of K-sized weakenings. Consider, for example, the theory $\Sigma = W_1 \cup \Omega_1$, where

$$\begin{array}{rcl} W_1 & = & \{\neg a \vee c,\ \neg b \vee c\} \\ \Omega_1 & = & \{a \vee b\}\ . \end{array}$$

Notice that all transformations in $\mathcal{T}^W[W_1, \Omega_1]$ are degenerate, as there are only two resolvents of elements in W_1 with elements in Ω_1 (*viz.*, $a \vee b$ and $b \vee c$) and neither is horn. As W_1 has no neighbors, AdComp cannot consider any alternative weakenings, meaning it will necessarily miss the superior weakening, $W_{opt} = \{c\}$.

However, notice that we could have reached this W_{opt} weakening in two steps, if we had used transformations that could produce new non-horn clauses. Given such transformations, we could then form the new pair $\langle W_2, \Omega_2\rangle$, where $W_2 = W_1$, and $\Omega_2 = \{\ a\vee b,\ b\vee c,\ a\vee c\ \}$. Now, by resolving the clauses in W_2 with those in Ω_2, we would produce the desired $W_3 = \{c\}$ (along with $\Omega_3 = \{\ b \vee c\ \}$).

Unfortunately, there is a basic problem with this approach: The score of any weakening/non-horn-complement pair $\langle W, \Omega\rangle$ depends only on the observed categoricity of the weakening part W (*i.e.*, on the values of $d(W,\sigma)$ used to approximate $\mathrm{D}[\,W\,]$). This means that the score of $\langle W, \Omega'\rangle$ is necessarily the same as the score of $\langle W, \Omega\rangle$, even though Ω' is different from Ω. Thus, $\langle W, \Omega'\rangle$ can never be strictly better than $\langle W, \Omega\rangle$, and so AdComp will never climb to it. This is why AdComp does not even generate these equal-cost neighbors.

There is an obvious alternative. Given $\langle W, \Omega\rangle$, the alternative AdComp$_1$ algorithm produces new non-horn components, $\{\Omega_i\}$, as well as new weakenings, $\{W_j\}$. Just like AdComp, this algorithm also compares the values of $d(W, Q)$ with each $d(W_j, Q)$, over a prescribed set of queries Q. If any $W' \in \{W_j\}$ passes the Equation 3 test, AdComp$_1$ will climb to this new weakening. Otherwise, if none of the alternative weakenings $\{W_j\}$ looks much better, AdComp$_1$ will randomly pick one of the alternative non-horn theories, $\Omega' \in \{\Omega_i\}$, and climb "sideways" to the weakening-pair $\langle W, \Omega'\rangle$. This produces a new neighborhood — a different set of neighboring weakenings $\{W'_j\}$ and of neighboring non-horn components $\{\Omega'_i\}$. AdComp$_1$ will then compare Ws score with each its neighbors, and climb to an $W'' \in \{W'_j\}$ if $d(W'', Q')$ is sufficiently better than $d(W, Q')$ for the (new) set of sample queries Q'. If none qualify, AdComp$_1$ will again walk side-ways, to one of the neighboring $\Omega'' \in \{\Omega'_i\}$; and so forth.

Of course, we may not want to wander about on this equal-score plateau forever. The AdComp$_2$ variant will permit only `MaxPlateauWalks` steps before terminating its search for a good weakening, where `MaxPlateauWalks` $\in \mathcal{Z}^+$ is a user-specified parameter. Another variant is AdComp$_3$: If none of the W_js appears better, AdComp$_3$ will stochastically decide whether to walk to a new $\Omega' \in \{\Omega_i\}$ (with probability `PlateauWalkProb`) or to terminate. Here, this `PlateauWalkProb` $\in [0,1]$ is a user-specified parameter.

Each of these three variants will have to prevent looping (*i.e.*, walking from Ω_1 to Ω_2 to ... and back to Ω_1), perhaps by imposing some ordering on the Ω_i theories, and only going from Ω_i to a new Ω_{i+1} with a strictly larger value. Also, each variant may use some bias on the set of Ω_is, to prefer some over others.

Extension 6. The AdComp process uses the function $K(\cdot)$ to bound the size of the weakening. In essence, this function quantifies how much time the user will allow the system to spend in answering a query, before insisting that it stop and return IDK. Hence, by selecting an appropriate $K(\cdot)$ function, the user can direct AdComp to the class of approximations that optimizes his implicit utility measure which embodies a particular tradeoff between efficiency and categoricity.

In general, the user may want to use a more general measure for ranking different approximations, which can depend on other factors as well. For example, is complete accuracy important? If not, how does it tradeoff with time concerns? Is incompleteness (in the form of using "IDK") better than errors? ...or are these two equally bad?

We can provide the user with greater flexibility by allowing him to specify his own scoring function, $c_i : \mathrm{Approx}_\infty(\Sigma) \times \mathcal{Q} \mapsto \Re$, where $c_i(\Upsilon,\sigma)$ indicates how well the approximation Υ does at solving σ.[12] Following [GE91], we allow this scoring function to be a combination of various factors, including accuracy, categoricity and efficiency. Given any such c_i, our goal is to find the performance system (call it PS_{c_i}) whose expected c_i score is maximal.

To illustrate the range of possible scoring functions, we can, for example, define c_0 to be a scoring function that imposes a very high penalty for incorrect or incomplete answers. If this penalty is sufficiently high, the optimal PS_{c_0} system will necessarily be sound and complete with respect to Σ, meaning it will have to use $\alpha = SND$. The AdComp system presented above implicitly uses a different function, call it c_1, that does not impose as severe a restriction: As it does not insist on completeness but does require (poly-time) efficiency, c_1 prefers performance systems that return IDK if the approximation does not cover the query. Hence, the c_1-optimal performance system uses $\alpha = IDK$ rather than $\alpha = SND$, in addition to imposing a limitation on the size of the approximations.

A slightly different criterion, encoded by the c_2 function, would explicitly use both computational time and

[12] The set $\mathrm{Approx}_\infty(\Sigma) = \bigcup_{k=1}^\infty \mathrm{Approx}_k(\Sigma)$ includes *all* horn approximations, of arbitrary size.

categoricity to rank approximations: $c_2(\langle S, W \rangle\ \sigma)$ is the time $PS = PS(S, W, \Sigma)$ spends answering σ if PS finds a definite answer, or a large negative value, B, if PS returns IDK. Here the size of B could indirectly determine, for example, whether $\alpha = IDK$ was better than $\alpha = SND$, and also the allowed size of the W set.

This suggests a way of addressing ISSUE 1 (WHICH α?): *viz.*, by using a variant, call it ADCOMP_α, that searches for a good α-function in parallel with its search for good weakenings and strengthenings. The particular c_i function specified will determine whether the optimal performance system should use $\alpha = SND$ or $\alpha = GUE$ or $\alpha = IDK$, or possibly some other pre-defined α function. (See [GS92].)

Future Work. First, we are currently implementing ADCOMP and plan to test it empirically in order to determine just how categorical and efficient it really is, both on real world problems and on "hard" cases [MSL92]. We also plan to compare ADCOMP with other theorem proving processes, including incomplete systems like GSAT [SLM92]. We anticipate that these studies will give us insights on many of the issues mentioned above, including the different ways of handling the "plateau problem" mentioned in Extension 5. Second, we are currently looking for a less sample-hungry variant of ADCOMP, perhaps based on a statistical stopping criterion that is less generous than the Chernoff bounds (Equation 4). A third extension is to find better sets of transformations, especially for moving about the space of horn weakenings. We will also consider other types of non-*Horn* approximations: syntactically defined formula from which derivation is efficient. The final task is to extend these ideas from propositional logic to full first-order predicate calculus.

4 Conclusion

It is often critical for an agent to use its knowledge to produce answers efficiently. Unfortunately, this task is intractable in general. [SK91] presents a way around this problem, describing an algorithm that produces a semantically similar Horn approximation from which many queries can be answered efficiently. Our paper extends that work by providing a more utilitarian objective function, one that prefers approximations that produce answers efficiently to most *anticipated* queries. It then defines the ADCOMP algorithm, that effectively finds near optimal approximations: ADCOMP uses a set of queries to estimate the distribution of anticipated queries, then uses this information to hill-climb in the space of possible approximations until reaching one that is, with high probability, near a local optimum. We also discuss various extensions to this algorithm, to permit it to use a more general user-specified utility measure, which can embody the user's tradeoffs between efficiency and accuracy, etc.

Our approach differs from earlier approaches to theory approximation in a number of significant ways: [1] ADCOMP uses a set of observed queries to "learn" an approximation that is essentially at a local optimum within the class of possible approximations; [2] this near-optimal approximation is guaranteed to perform tractable inference; and [3] ADCOMP learns this near-optimal approximation without computing answers to queries using original theory, meaning it can find effective approximations efficiently.

A Proof of Theorem 1

Theorem 1 *The $\text{ADCOMP}(\Sigma, \epsilon, \delta, K)$ process incrementally produces a series of weakenings $\langle W_0, W_1, \ldots, W_m \rangle$ and (independently) a sequence of strengthenings $\langle S_0, S_1, \ldots, S_n \rangle$ such that, with probability at least $1 - \delta$,*

1. *each successive approximation has an expected score that is strictly better than its predecessor's; i.e.,*
$$\begin{aligned} \mathcal{C}[\,\langle S_{i+1}, W_j \rangle\,] &> \mathcal{C}[\,\langle S_i, W_j \rangle\,] \\ \mathcal{C}[\,\langle S_i, W_{j+1} \rangle\,] &> \mathcal{C}[\,\langle S_i, W_j \rangle\,] \end{aligned}$$
and
2. *the final approximation $\langle S_n, W_m \rangle$ is an ϵ-local optimum; i.e., its expected score is within ϵ of the best expected score among its neighbors:*
$$\forall\ \tau \in \mathcal{T}^S: \quad \mathcal{C}[\,\langle S_n, W_m \rangle\,] \ \geq\ \mathcal{C}[\ \langle \tau(S_n), W_m \rangle\ ] - \epsilon$$
$$\forall\ \tau \in \mathcal{T}^W: \quad \mathcal{C}[\,\langle S_n, W_m \rangle\,] \ \geq\ \mathcal{C}[\ \langle S_n, \tau(W_m) \rangle\ ] - \epsilon.$$

Moreover, ADCOMP requires only polynomial time (and hence only a polynomial number of samples) to decide whether to move from S_i to S_{i+1} (resp., from W_j to W_{j+1}) or terminate with a final S_n (resp., with a final W_m). $\Box$.

Proof: Subsection 2.6 above already established ADCOMP's computational efficiency.

To prove parts 1 and 2 of the theorem: Consider first a single iteration of the ADCOMP algorithm, and consider only the strengthenings. Notice there are two ways that ADCOMP can make a mistake:

1. If some $S' \in \text{NEIGHS}$ appears to be better than S_j but is not; or
2. If some $S' \in \text{NEIGHS}$ is really more than ϵ better than S_j, but appears not to be.

Let

$$p_1^j \;=\; Pr\left[\begin{array}{r} \exists S' \in \text{NEIGHS}.\ d(S_j,\, Q_j) - d(S',\, Q_j) \geq \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[\,S_j\,] < \mathrm{D}[\,S'\,] \end{array}\right]$$
$$p_2^j \;=\; Pr\left[\begin{array}{r} \exists S' \in \text{NEIGHS}.\ d(S_j,\, Q_j) - d(S',\, Q_j) < \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[\,S'\,] < \mathrm{D}[\,S_j\,] - \epsilon \end{array}\right]$$

be the respective probabilities of these events. Now observe that

$$\begin{aligned} p_1^j &\leq \sum_{S' \in NeighS} Pr\left[\begin{array}{r} d(S_j,\, Q_j) - d(S',\, Q_j) \geq \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[\,S'\,] - \mathrm{D}[\,S_j\,] < 0 \end{array}\right] \\ &\leq \sum_{S' \in NeighS} e^{-2 n_j \left(\frac{\epsilon}{2}\right)^2} \qquad (5) \\ &\leq |\text{NEIGHS}|\, e^{-2\left(\frac{2}{\epsilon^2} \ln \frac{2\, j^2\, \pi^2\, \max\{|NeighS|,\, |NeighW|\}}{3\,\delta}\right)\left(\frac{\epsilon}{2}\right)^2} \\ &= |\text{NEIGHS}| \frac{3\,\delta}{2\, j^2\, \pi^2 \max\{|NeighS|,\, |NeighW|\}} \\ &\leq \frac{1}{j^2} \frac{3\,\delta}{2\,\pi^2} \end{aligned}$$

Line 5 uses Chernoff bounds (Equation 4).[13] Similarly,

$$\begin{aligned} p_2^j \quad &\leq \quad \sum_{S' \in NeighS} Pr\left[\begin{array}{r} d(S_j,\, Q_j) - d(S',\, Q_j) < \frac{\epsilon}{2} \\ \text{and } \mathrm{D}[\,S_j\,] - \mathrm{D}[\,S'\,] > \epsilon \end{array} \right] \\ &\leq \quad \sum_{S' \in NeighS} e^{-2n_j\left(\frac{\epsilon}{2}\right)^2} \quad \leq \quad \frac{1}{j^2}\frac{3\,\delta}{2\,\pi^2} \end{aligned}$$

Hence, the probability of ever making either mistake at any iteration is under

$$\begin{aligned} \sum_{j=1}^{\infty} p_1^j + p_2^j \quad &\leq \quad \sum_{j=1}^{\infty} \frac{1}{j^2}\frac{3\,\delta}{2\,\pi^2} + \frac{1}{j^2}\frac{3\,\delta}{2\,\pi^2} \\ &= \quad \delta\frac{3}{\pi^2}\sum_{j=1}^{\infty}\frac{1}{j^2} \quad = \quad \delta\frac{3}{\pi^2}\frac{\pi^2}{6} \quad = \quad \frac{\delta}{2} \end{aligned}$$

The same arguments, *mutatis mutandis*, hold for finding good weakening: the probability of either climbing to an inferior weakening, or stopping at a weakening that is not an ϵ-local optimum, is also bounded by $\delta/2$. Hence, the probability of either making a mistake for the strengthenings, or weakenings, is under $\frac{\delta}{2} + \frac{\delta}{2} = \delta$, as desired. $\Box$.

References

[Bol85] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[Che52] 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.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC71*, pages 151–58, 1971.

[DB88] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54, August 1988.

[DE92] Mukesh Dalal and David Etherington. Tractable approximate deduction using limited vocabulary. In *Proceedings of CSCSI-92*, Vancouver, May 1992.

[DG84] 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.

[EBBK89] David W. Etherington, Alex Borgida, Ronald J. Brachman, and Henry Kautz. Vivid knowledge and tractable reasoning: Preliminary report. In *Proceedings of IJCAI-89*, pages 1146–52, 1989.

[GE91] Russell Greiner and Charles Elkan. Measuring and improving the effectiveness of representations. In *Proceedings of IJCAI-91*, pages 518–24, Sydney, Australia, August 1991.

[13] This relies on the fact that the distribution of queries is stationary, meaning that, for any given pair of strengthenings S_j and S', the values of the random variables $\Delta_i = d(S_j,\, \sigma_i) - d(S',\, \sigma_i)$ are drawn from a stationary distribution.

[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.

[GJ92] Russell Greiner and Igor Jurišica. A statistical approach to solving the EBL utility problem. In *Proceedings of AAAI-92*, San Jose, 1992.

[GS92] Russell Greiner and Dale Schuurmans. Learning useful horn approximations. Technical report, Siemens Corporate Research, 1992.

[KS92] Henry Kautz and Bart Selman. Speeding inference by acquiring new concepts. In *Proceedings of AAAI-92*, San Jose, July 1992.

[MSL92] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distribution of sat problems. In *Proceedings of AAAI-92*, San Jose, July 1992.

[SK91] Bart Selman and Henry Kautz. Knowledge compilation using horn approximations. In *Proceedings of AAAI-91*, pages 904–09, Anaheim, August 1991.

[SLM92] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, pages 440–46, San Jose, July 1992.