Appears in
Artificial Intelligence, 84:1–2 (July 1996), p. 177–204.

PALO: A Probabilistic Hill-Climbing Algorithm*

Russell Greiner†
Siemens Corporate Research, Princeton, NJ 08540
greiner@scr.siemens.com (609) 734-3627

Abstract

Many learning systems search through a space of possible performance elements, seeking an element whose expected utility, over the distribution of problems, is high. As the task of finding the globally optimal element is often intractable, many practical learning systems instead hill-climb to a local optimum. Unfortunately, even this is problematic as the learner typically does not know the underlying distribution of problems, which it needs to determine an element's expected utility. This paper addresses the task of approximating this hill-climbing search when the utility function can only be estimated by sampling. We present a general algorithm, PALO, that returns an element that is, with provably high probability, essentially a local optimum. We then demonstrate the generality of this algorithm by presenting three distinct applications, that respectively find an element whose efficiency, accuracy or completeness is nearly optimal. These results suggest approaches to solving the utility problem from explanation-based learning, the multiple extension problem from nonmonotonic reasoning and the tractability/completeness tradeoff problem from knowledge representation.

1 Introduction

Many learning tasks can be viewed as a search through a space of possible performance elements seeking an element that is optimal, based on some utility measure. As examples, inductive systems seek classifiers whose classifications are optimally accurate, and many explanation-based learning [17, 60] and chunking [55] systems seek problem solvers that are optimally efficient [59, 30]. In each of these cases, the utility function used to compare the different elements (*e.g.*, classifiers or problem solvers) is defined as the expected value of

*This paper expands the short article, "Probabilistic Hill-Climbing: Theory and Applications" that was awarded the "Artificial Intelligence Journal Best Paper Award" at the *Ninth Canadian Conference on Artificial Intelligence* (CSCSI-92), in Vancouver, in May 1992.

†This work began at the University of Toronto, where it was supported in part by the Institute for Robotics and Intelligent Systems and by an operating grant from the National Science and Engineering Research Council of Canada. I gratefully acknowledge receiving many helpful comments from William Cohen, Dale Schuurmans and the anonymous referees.

a particular scoring function, averaged over the distribution of samples (or goals, queries, problems, ...) that will be seen [42, 38].

There are at least two problems with implementing such a learning system: First, we need to know the distribution of samples to determine which element is optimal; unfortunately, this information is usually unknown. There are, of course, standard statistical techniques that use the set of observed samples to estimate the needed information; and several classes of learning systems have incorporated these techniques. For example, many "PAC-learning" systems [78] use these estimates to identify an element that is, with high probability, approximately a global optimum.

This leads to the second problem: unfortunately, the task of identifying the globally optimal element, even given the correct distribution information, is intractable for many spaces of elements [30, 42]. A common response is to build a system that hill-climbs towards a *local* optimum. Many well-known inductive learning systems, including BACKPROP [44], genetic algorithms [6] and C4.5 [68], use this approach, as do many speedup learning methods; see especially [28]. Unfortunately, few existing systems guarantee that each hill-climbing step is even an improvement, meaning the final element is not always even superior to the initial one, much less an optimum in the space of elements. Moreover, fewer systems include a stopping criterion to determine when the learning has reached a point of diminishing returns.

The work presented here draws ideas from both of these themes: In particular, it describes a general learning algorithm, PALO, that hill-climbs to a local optimum, using a utility measure that is estimated by sampling. Given any parameters $\epsilon, \delta > 0$, PALO efficiently produces an element whose expected utility is, with probability at least $1 - \delta$, an ϵ-local optimal.[1] As PALO processes one sample at a time, it is an incremental (as opposed to "batched") learner, which uses statistical tests to mollify the effects of the sample order. Moreover, this system can often work unobtrusively [62], passively gathering the statistics it needs by simply watching a performance element solve problems relevant to a user's applications. Here, the incremental cost of PALO's hill-climbing, over the cost of simply solving performance problems, can be very minor.

Section 2 first compares and contrasts our approach with others from the literature. Section 3 motivates the use of "expected utility" as a quality measure for comparing performance elements. Section 4 then defines the general PALO algorithm, which deals sequentially with a series of performance elements $\Theta_1, \ldots, \Theta_m$ such that, with high confidence, each Θ_{i+1} is an improvement over Θ_i and the performance of the final Θ_m is a local optimum in the space being searched. It also describes the statistical tool used to determine whether the result of a proposed modification is better than the original performance element; this tool can be viewed as a mathematically rigorous version of Minton's "utility analysis" [59]. Section 5 demonstrates the generality of our approach by presenting three different applications of the PALO system, each using its own set of transformations to find a near-optimal element within its particular set of performance elements, where optimality is defined in terms of efficiency, accuracy or completeness, respectively. It also summarizes an empirical study, to illustrate PALO's behavior in a particular situation. Section 6 discusses several variations, extensions and limitations, of our approach. The appendix contains proofs of the claims made in the

[1] Theorem 1 below defines both our sense of efficiency, and "ϵ-local optimality".

paper.

2 Related Results

Finding an Element with the Best Average Performance: There are several other projects that use statistical techniques to find a performance element whose average performance is optimal. In general, each such learning system must evaluate each of N performance elements for each of k training samples. Maron and Moore [58] describe a system that works when there is a relatively small, and explicit, set of elements, meaning the $N \times k$ "element–sample evaluations" can be performed explicitly. Their "Hoeffding Race" approach attempts to reduce the total number of element–sample evaluations by removing an element as soon as it is statistically clear that this element will not be the optimum.

Fong [22] presents a different, more mathematically rigorous, solution to the problem of reducing the number of element–sample evaluations, by specifying which single element should deal with each sample. The resulting "γ-IE" framework extends Kaelbling's IE system [50].

Combinatorial Space $\Rightarrow$ Hill-Climbing: These approaches work when there is an explicit representation of all possible performance elements. In many cases, however, there are an implicitly-defined combinatorial number of elements. Here, it makes sense to impose a "structure" on the space by connecting each element to the set of its neighbors (which form a small subset of the space), and then use a hill-climbing system to climb successively from the "current element" to one of its neighbors. There are, of course, a huge number of such hill-climbing systems used throughout machine learning, as well as almost every other field of computer science. Each such system must evaluate the currently-proposed performance element against its neighbors. This comparison is trivial if each element's "quality" is easily computed. This is not true in our case, as our quality measure is the expected value of the element's score on an instance, which is not known as it depends on the *unknown* distribution of instances.

As mentioned above, many learning systems attempt to address this challenge, of finding an element whose expected behavior is optimal; see for example the learning procedures used by symbolic classifiers [68], neural nets [44] and genetic algorithms [6]. Each of these systems uses a set of training samples to estimate the distribution, then uses this information to determine when one element is superior to another. Most systems do this implicitly, and heuristically. By contrast, our PALO system performs an *explicit* statistical test to determine, *with prescribed confidence*, when one element is superior to another.

As such, it is very similar to the COMPOSER system of Gratch, deJong and Chien [28, 29, 27]. COMPOSER differs from PALO in two significant ways. First, COMPOSER will use all available samples when hill-climbing. By contrast, PALO will stop and return the currently-best performance element Θ if none of Θ's neighbors appears significantly better than Θ — which means PALO will stop on reaching a point of "diminishing returns". The second difference is in the statistical criteria used: While PALO's tests (based on Hoeffding's inequality; Section 4's Equation 6) holds for for any bounded stationary distribution of samples, COMPOSER's test (based on Nádas rule) implicitly assumes that the samples are drawn from a normal distribution. While the COMPOSER assumption is standard, and empirically is often appropriate, it is not guaranteed to be correct; see Section 6.2. Second, in the interest of producing an empirically effective system, COMPOSER also makes other simplifying assumptions; for example, it does not make PALO's conservative (but mathematically necessary) assumption that the errors on successive hill-climbs will add.[2]

"Rationality": Doyle and Patil [19] recently argued against the standard practice of using worst case analyses to decide amongst different possible performance elements, and instead advocated using an expected case analysis. This raises the obvious questions of how to obtain the distribution information required to determine the expected case behavior, and then, what to do with such information, once it is available. We view our approach, as embodied in the PALO system, as addressing exactly those questions.

Moreover, our PALO exhibits a type of "Type II rationality" [26], as it seeks an element whose expected utility is optimal, *subject to the resource constraint* of spending only a feasible amount of time to find such an element. Many other systems are similarly motivated by this issue of computational effectiveness — *i.e.*, what is the best performance the system can exhibit, given only limited computational resources; *cf.*, the works by Horovitz [46], Etzioni [21] and Russell, Subramanian and Parr [70]. These other systems, however, either assume that the utilities of the various elements are immediately available, or supply statistical methods for estimating these utilities for *all* of the elements (akin to the Hoeffding Race and γ-IE framework mentioned above). By contrast, PALO will only estimate the utilities for the elements that can be reached in the current phase of the hill-climbing process. As this is a small subset of the total space of elements, PALO can reach statistically significant conclusions after observing *many fewer* samples than are required by the other systems that must estimate the utilities of *all* of the elements; this allows PALO to begin climbing long before the other algorithms have finished collecting samples. Moreover, this also means that PALO will usually require a smaller total number of samples; see Note N-PALO5 in Section 4.

Incremental Algorithms: Many other methods attempt to make effective use of the training samples; *cf.*, reinforcement learning algorithms [50, 76] and systems that address the "bandit problem" [3, 64]. Each of these systems also makes a sequence of decisions, attempting to maximize its total reward. Such systems tend to run continuously, making successive decisions; and they are often evaluated in terms of the number of mistakes they make over their entire learning+performance lifetimes. By contrast, our PALO is evaluated only in terms of the quality of the performance element it returns, provided it returns this answer within a reasonable — "polynomial" — amount of time.

All of these other incremental learners climb through a series of different elements. PALO, however, is probabilistically guaranteed to improve over time: *i.e.*, the performance element produced at the j^{th} hill-climbing step is, with high probability, strictly better than the previous one, produced on the $j-1^{st}$ iteration. As such, it also resembles *anytime algorithms* [4, 16], but differs from standard anytime algorithms by terminating on reaching a

[2] A less major difference is that the COMPOSER system, like the Hoeffding Race and γ-IE systems mentioned above, qualifies as a "wrapper" learner [48, 10], as COMPOSER views each performance element as a black box, whose behavior can be sampled, but whose internals are unavailable. By contrast, PALO will sometimes examine the internals of the performance elements, and use this structural information to efficiently determine the scores of the neighboring elements; see Section 5.1 and [36].

point of diminishing returns.

"Probabilistic Hill-climbing": Finally, as the phrase "probabilistic hill-climbing" may suggest "simulated annealing" [53] to many readers, it is worth explicitly distinguishing these different ideas: The general simulated annealing process assumes that the quality measure used to compare different elements is accurate; its use of "probabilistic" refers to the stochastic way in which simulated annealing probabilistically decides whether to climb "downhill", in an attempt to avoid local optima. By contrast, our PALO system does not know the quality of each element, and so must estimate these values. Our use of "probabilistic" refers to the uncertainty of these estimates, due to possible sampling error. One could, of course, write a PALO-like that used an annealing process to climb downhill occasionally; however, this system would not satisfy the useful specifications presented in Theorem 1.

3 Framework

To illustrate the relevant concepts, consider the following example: A pure PROLOG program will return a set of answers to each query posed [12]. We can form new programs by rearranging the order of the clauses. While these programs will return the same set of answers, they can require differing amounts of time to find the first answer to each query. Our goal is to determine the "optimally efficient program"; *i.e.*, the ordering of the clauses that requires the minimal average time to find an answer, over the distribution of queries.

In general, we assume as given a (possibly infinite) set of performance elements $\mathcal{S}_\Theta = \{\Theta_i\}$, where each $\Theta \in \mathcal{S}_\Theta$ is a system that returns an answer to each given problem (or query or goal) $q_i \in \mathcal{Q}$, where $\mathcal{Q} = \{q_1, q_2, \ldots\}$ is the set of all possible problems.[3] We also use a given utility function $c\colon \mathcal{S}_\Theta \times \mathcal{Q} \mapsto \Re$, where $c(\Theta,\, q)$ measures how well the element Θ does at solving the problem q. In the PROLOG context, each performance element Θ corresponds to an ordering of the clauses, $\mathcal{S}_\Theta$ is the set of these PROLOG programs, each sample problem is a query, and $c(\Theta,\, q)$ quantifies the time Θ requires to solve q. (Section 5 later defines other classes of performance elements, samples and utility functions.) We insist that $c(\cdot,\, \cdot)$ have a bounded range of possible values $\lambda = \lambda(c, \mathcal{Q}, \mathcal{S}_\Theta) \in \Re^+$; *i.e.*,

$$\forall \Theta \in \mathcal{S}_\Theta,\ \forall q \in \mathcal{Q}\colon\ \min(\Theta)\ \le\ c(\Theta,\, q)\ \le\ \min(\Theta) + \lambda\ . \tag{1}$$

where $\min(\Theta) = \min_{q \in \mathcal{Q}}\{c(\Theta,\, q)\}$ is the minimal utility value of Θ, over all instances $q \in \mathcal{Q}$.

This utility function specifies the score of the performance element Θ_i on an individual instance. Our performance elements, however, will have to solve an entire ensemble of problems. To specify which element is best overall, we must therefore consider the distribution of problems that our performance elements will encounter. We model this using a stationary probability function, $Pr\colon \mathcal{Q} \mapsto [0, 1]$, where $Pr[\, q_i\,]$ denotes the probability that the problem q_i is selected. We then define the *expected utility* of a performance element in the obvious

[3] We assume that $\mathcal{Q}$ is countable for purely pedagogical reasons. There are obvious ways of extending our analysis to handle an uncountably infinite set of problems as well.

way:

$$\mathrm{C}(\,\Theta\,) \ \stackrel{def}{=}\ E_{q \in Pr}[\,c(\Theta,\,q)\,] \ =\ \sum_{q \in \mathcal{Q}} Pr[\,q\,] \times c(\Theta,\,q) \tag{2}$$

Our underlying challenge is to find the performance element whose expected utility is maximal. As mentioned above, there are two problems: First, the problem distribution, needed to determine which element is optimal, is usually unknown. Second, even if we knew that distribution information, the task of identifying the optimal element is often intractable.

4 The PALO_1 Algorithm

This section presents a learning system, "PALO_1",[4] that side-steps the above problems by using a set of sample queries to estimate the distribution, and by hill-climbing efficiently from a given initial Θ_1 to a performance element that is, with high probability, essentially a local optimum. This section first states the fundamental theorem that specifies PALO_1's functionality, then summarizes PALO_1's code and finally sketches the motivation underlying the theorem.

As shown in Figure 1, PALO_1 takes as arguments an initial performance element $\Theta_1 \in \mathcal{S}_\Theta$, a collection of possible transformations $\mathcal{T} = \{\tau_j\}$, where each $\tau_j : \mathcal{S}_\Theta \mapsto \mathcal{S}_\Theta$ maps one performance element to another, and parameters $\epsilon, \delta > 0$, that bound the allowed error and confidence. In the context of our PROLOG example, each τ_i rearranges the order of a pair of clauses; *e.g.*, $\tau_{3,1}(\Theta)$ moves Θ's third clause to the beginning.

PALO_1 uses an oracle that draws samples from $\mathcal{Q} = \{q_j\}$ at random, according to the fixed (but unknown) distribution $Pr[\cdot]$. (Here, each sample is a query, which could be posed by the user of the performance system; see Note N-PALO1 below.) Under specified conditions, PALO_1 will climb from the given initial performance element Θ_1 to one of Θ_1's neighbors, $\Theta_2 = \tau_1(\Theta_1)$ for some $\tau_1 \in \mathcal{T}$, and then possibly from this Θ_2 to a different $\Theta_3 = \tau_2(\Theta_2)$, and so on, until (eventually) reaching a Θ_m, which is returned.

Theorem 1 specifies PALO_1's behavior. It uses

$$\mathcal{T}[\Theta] \quad = \quad \{\tau(\Theta) \in \mathcal{S}_\Theta \mid \tau \in \mathcal{T} \ \&\ \tau(\Theta) \neq \Theta\}$$

which is the set of Θ's neighbors. (The proof for this theorem appears in the appendix.)

Theorem 1 *The* $\text{PALO}_1(\,\Theta_1,\,\mathcal{T},\,\epsilon,\,\delta\,)$ *process incrementally produces a series of performance elements* $\Theta_1, \Theta_2, \ldots, \Theta_m$*, such that each* $\Theta_{j+1} = \tau_j(\Theta_j)$ *using some* $\tau_j \in \mathcal{T}$ *and, with probability at least* $1 - \delta$*,*

1. *the expected utility of each performance element is strictly better than its predecessors i.e.,*
$$\forall 1 \leq i < j \leq m : \ C(\,\Theta_j\,)\ >\ C(\,\Theta_i\,); \quad \textit{and}$$
2. *the final performance element returned by* PALO_1, Θ_m*, is an "*ϵ*-local optimum" — i.e.,*
$$\neg\exists \tau \in \mathcal{T} : \quad C(\ \tau(\Theta_m)\ )\ \geq\ C(\,\Theta_m\,) + \epsilon\ .$$

[4] "PALO" abbreviates "<u>P</u>robably <u>A</u>pproximately <u>L</u>ocally <u>O</u>ptimal". Section 6.1 explains the "1" subscript.

Algorithm $\textsc{palo}_1(\,\Theta_1,\,\mathcal{T},\,\epsilon,\,\delta\,)$

  For $j = 1..\infty$ do

    Let $\delta_j \;\leftarrow\; \frac{6\;\delta}{j^2\;\pi^2}$,

      $\mathcal{T}[\Theta_j] \;\leftarrow\; \{\tau(\Theta_j) \in \mathcal{S}_\Theta | \tau \in \mathcal{T} \;\&\; \tau(\Theta_j) \neq \Theta_j\}$, *% $\mathcal{T}[\Theta_j]$ are Θ_j's neighbors*

      $L_j \;\leftarrow\; \left\lceil 2\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \ln\frac{2\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right\rceil$ *% L_j is max # of samples on j^{th} iteration* $\langle L1\rangle$

    ForEach $\Theta' \in \mathcal{T}[\Theta_j]$ **do** **Let** $\Delta(\,\Theta_j,\,\Theta',\,0\,) \;\leftarrow\; 0$.

    For $i = 1..L_j$ do *% Get and process i^{th} query*

      Get sample q_i (from oracle)

      ForEach $\Theta' \in \mathcal{T}[\Theta_j]$ **do**

        Let $\Delta(\,\Theta_j,\,\Theta',\,i\,) \;\leftarrow\; \Delta(\,\Theta_j,\,\Theta',\,i{-}1\,) \;+\; [c(\Theta',\,q_i) - c(\Theta_j,\,q_i)]$ $\langle L2\rangle$

      If $i < L_j$

      Then

        If $\exists\Theta' \in \mathcal{T}[\Theta_j]$ s.t. $\frac{1}{i}\Delta(\,\Theta_j,\,\Theta',\,i\,) \;>\; \Lambda(\Theta_j,\Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2\,(L_j-1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ $\langle L3\rangle$

        Then **Let** $\Theta_{j+1} \leftarrow \Theta'$

          Exit For (inner loop)

        Else If $\forall\Theta' \in \mathcal{T}[\Theta_j]$: $\frac{1}{i}\Delta(\,\Theta_j,\,\Theta',\,i\,) \;<\; \epsilon - \Lambda(\Theta_j,\Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2\,(L_j-1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ $\langle L4\rangle$

          Then **Return** Θ_j (Exiting both inner and outer For Loops)

      Else

        If $\exists\Theta' \in \mathcal{T}[\Theta_j]$ s.t. $\frac{1}{L_j}\Delta(\,\Theta_j,\,\Theta',\,L_j\,) \;>\; \frac{\epsilon}{2}$ $\langle L5\rangle$

        Then **Let** $\Theta_{j+1} \leftarrow \Theta'$

          Exit For (inner loop)

        Else **Return** Θ_j (Exiting both inner and outer For loops)

    End For (inner loop)

  End For (outer loop)

End $\textsc{palo}_1$

Figure 1: Code for $\textsc{palo}_1$ Algorithm

Moreover, $\textsc{palo}_1$ *will stay at any* Θ_j *(before either terminating or climbing to a new* Θ_{j+1}*) for a number of samples that is polynomial in* $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $\lambda(c, \mathcal{Q}, \mathcal{S}_\Theta)$ *and* $|\mathcal{T}[\Theta_j]|$, *and will terminate with probability* 1, *provided* $|\mathcal{S}_\Theta|$ *is finite.* $\Box$.

The $\textsc{palo}_1$ code, shown in Figure 1, uses two additional terms: For any pair of performance elements $\Theta, \Theta' \in \mathcal{S}_\Theta$,

$$\Lambda(\Theta,\Theta') \;=\; \max_{q\in\mathcal{Q}}\{\, c(\Theta',\,q) - c(\Theta,\,q)\,\} \;-\; \min_{q\in\mathcal{Q}}\{\, c(\Theta',\,q) - c(\Theta,\,q)\,\} \tag{3}$$

bounds the range of possible values for $c(\Theta',\,q) - c(\Theta,\,q)$ over all samples $q \in \mathcal{Q}$; and

$$\Lambda[\Theta] \;=\; \max_{\Theta'\in\mathcal{T}[\Theta]}\{\,\Lambda(\Theta,\Theta')\,\} \tag{4}$$

is the largest such range for a given Θ, over all neighbors $\Theta' \in \mathcal{T}[\Theta]$. Notice that $\Lambda[\Theta] \leq 2\lambda$ for any Θ, using the λ specified in Equation 1.

To summarize the code: PALO$_1$ examines a sequence of sample queries, one by one. On seeing each query, PALO$_1$ computes the utility of the given Θ_1 performance element, summed over all of the queries seen so far, and compares that value with comparable values for each of Θ_1's neighbors (Line $\langle L2 \rangle$). If any neighbor appears to be significantly better (Line $\langle L3 \rangle$), it becomes the new performance element Θ_2. PALO$_1$ then compares Θ_2's performance with that of Θ_2's neighbors over the next set of samples; and once again, if any of Θ_2's neighbors appears much better, PALO$_1$ will climb to this apparently-superior element Θ_3, and so forth. On the other hand, if all of Θ_j's neighbors appear comparable to or worse than the current Θ_j (Line $\langle L4 \rangle$), PALO$_1$ will terminate, returning Θ_j. If neither of these conditions holds, PALO$_1$ will, in general, simply process the next query, then use this query, in addition to the previous ones, when comparing the current Θ_j to its neighbors. However, if PALO$_1$ has dealt with this current Θ_j for a sufficiently large number of queries (L_j, from Line $\langle L1 \rangle$), PALO$_1$ will use easier-to-satisfy thresholds to decide whether to climb to some Θ_{j+1} or terminate (Line $\langle L5 \rangle$), and will necessarily perform one of those actions.

Notice PALO$_1$ climbs from Θ to a new $\Theta' = \tau(\Theta)$ if Θ' is likely to be better than Θ; *i.e.*, if we are highly confident that $\mathrm{C}(\,\Theta'\,) > \mathrm{C}(\,\Theta\,)$, or equivalently, if

$$\mu \stackrel{def}{=} \mathrm{C}(\,\Theta'\,) - \mathrm{C}(\,\Theta\,) \quad > \quad 0\,. \tag{5}$$

Unfortunately, as this $\mathrm{C}(\,\Theta'\,) - \mathrm{C}(\,\Theta\,)$ quantity depends on the unknown distribution, we cannot immediately determine if Equation 5 holds. We can, however, obtain an approximation that usually is good enough. To do this, define the random variable

$$d_i \quad \stackrel{def}{=} \quad c(\Theta',\, q_i) - c(\Theta,\, q_i)$$

to be the difference in utility between using Θ' to deal with the query q_i, versus using Θ. As each query q_i is selected randomly according to the fixed distribution, these d_is are independent, identically distributed random variables whose common mean is Equation 5's μ.

Now let $Y_n \stackrel{def}{=} \frac{1}{n}\sum_{j=1}^{n} d_i$ be the sample mean over n samples. (Notice the $\Delta(\,\Theta,\,\Theta',\,n\,)$ quantity computed on Line $\langle L2 \rangle$, corresponds to $n \times Y_n$.) From the Law of Large Numbers, we know that this average will tend to the true population mean μ as $n \to \infty$; *i.e.*, $\mu = \lim_{n\to\infty} Y_n$. Hoeffding's inequality [45, 11] bounds the probable rate of convergence: the probability that "Y_n is more than $\mu + \gamma$" goes to 0 exponentially fast as n increases; and, for a fixed n, exponentially as γ increases. Formally,[5]

$$Pr[\,Y_n > \mu + \gamma\,] \;\le\; e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2} \qquad Pr[\,Y_n < \mu - \gamma\,] \;\le\; e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2} \tag{6}$$

where $\Lambda = \Lambda(\Theta, \Theta')$ is the range of possible values of $c(\Theta',\, q_i) - c(\Theta,\, q_i)$ defined in Equation 3.

The PALO$_1$ algorithm uses these equations and the values of $\Delta(\,\Theta_j,\,\Theta',\,i\,)$ to determine both how confident we should be that $\mathrm{C}(\,\Theta'\,) > \mathrm{C}(\,\Theta_j\,)$ (Lines $\langle L3 \rangle$ and $\langle L5 \rangle$) and whether

[5] See [5, p. 12]. *N.b.*, these inequalities do *not* require that the underlying distributions be normal; instead, they hold for any arbitrary bounded distribution.

any "$\mathcal{T}$-neighbor" of Θ_j (*i.e.*, any $\tau_k(\Theta_j)$) is more than ϵ better than Θ_j (Lines $\langle L4\rangle$ and $\langle L5\rangle$); see the proof in the appendix.

We close this section with five general comments on the PALO$_1$ framework and algorithm.[6]

N-PALO1. The samples that PALO$_1$ uses may be produced by a user of the performance system, who is simply asking questions relevant to her current applications; here, PALO$_1$ is unobtrusively gathering statistics as the user is solving her own problems [62]. This means that the total cost of the overall system, that both solves performance problems and learns by hill-climbing to successive performance elements, can be only marginally more than the cost of only running the performance element to simply solve the performance problems.

PALO uses these user-provided samples as its objective is to approximate the average utility values of the elements, over the distribution of problems that the performance element will actually address. This "average case analysis" differs from several other approaches as, for example, we are not assuming that this distribution of problems will be uniform [25], nor that it will necessarily correspond to any particular collection of "benchmark challenge problems" [52].

N-PALO2. A "0-local optimum" corresponds exactly to the standard notion of local optimum; hence our "ϵ-local optimum" (condition 2 of Theorem 1) generalizes local optimality. This means that PALO$_1$'s output Θ_m will (with high probability) be a real local optimum if the difference in utility between every two distinct performance elements, Θ and $\tau(\Theta)$, is always larger than ϵ. Thus, for sufficiently small values of ϵ, PALO$_1$ will produce a *bona fide* local optimum.

N-PALO3. As $\mathcal{T}[\Theta]$ is the *set* of distinct, non-degenerate $\tau(\Theta)$ elements, it can be much smaller than $|\mathcal{T}|$, as there can be many $\tau \in \mathcal{T}$ that map Θ_j to itself, and many other disjoint pairs $\langle \tau, \tau' \rangle$ such that $\tau(\Theta_j) = \tau'(\Theta_j)$.

N-PALO4. Our PALO$_1$ will (probably) process more samples using later performance elements than using the earlier ones, as its tests (directly associated with Lines $\langle L3\rangle$ and $\langle L4\rangle$, and indirectly with Line $\langle L1\rangle$) are increasingly more difficult to pass. This behavior is desirable, as it means that the overall system is dealing with larger numbers of samples using later, and therefore better, elements.

N-PALO5. The PALO$_1$ systems uses different sets of samples on each climb. An alternative approach, which we call PALO$_{all}$, will instead obtain a *single* set of samples at the start, and use them to (simultaneously) estimate the utilities of *all* of the performance elements — in the manner suggested by some of the systems discussed in Section 2. In particular, it will use these estimates when comparing different elements during the hill-climbing process. If the total space contains N different elements, PALO$_{all}$ will have to simultaneously estimate

[6] We will end many sections with such comments. In each case, the casual reader may skip them, and suffer no loss of continuity.

all N elements to within $\pm\epsilon/2$, with probability at least $1-\delta$; this requires

$$M_{all} \;=\; \frac{1}{2}\left(\frac{\Lambda_{max}}{\epsilon/2}\right)^2 \ln\frac{2N}{\delta} \;=\; 2\left(\frac{\Lambda_{max}}{\epsilon}\right)^2 \left[\ln N \;+\; \ln\frac{2}{\delta}\right] \tag{7}$$

samples, where $\Lambda_{max} = \max_j \Lambda[\Theta_j]$ is the largest range of values, over all performance elements Θ_j.

In most situations, PALO$_1$ will require *fewer* samples than PALO$_{all}$. To see this, note that PALO$_1$ requires at most L_j samples to deal with the j^{th} element, meaning it can perform $k-1$ climbs, and terminate, using *at most*

$$\begin{aligned} m_{all} \;&=\; \sum_{j=1}^{k} L_j \;=\; \sum_{j=1}^{k} 2\left(\frac{\Lambda[\Theta_j]}{\epsilon}\right)^2 \ln\frac{2\,|\mathcal{T}[\Theta_j]|}{\delta_j} \\ &\le\; 2\left(\frac{\Lambda_{max}}{\epsilon}\right)^2 \sum_{j=1}^{k} \ln\frac{2\,j^2\,\pi^2|\mathcal{T}[\Theta_j]|}{6\delta} \;=\; 2\left(\frac{\Lambda_{max}}{\epsilon}\right)^2 \sum_{j=1}^{k}\left(\ln|\mathcal{T}[\Theta_j]| \;+\; \ln\frac{j^2\,\pi^2}{3\delta}\right) \end{aligned} \tag{8}$$

samples. In practice, PALO$_1$ will usually require far fewer samples than this to deal with its k elements as (1) $\Lambda[\Theta_j]$ is often far under Λ_{max}, and also (2) PALO$_1$ will usually use far fewer than L_j samples on its j^{th} element, thanks to its Line $\langle L3\rangle$ and Line $\langle L4\rangle$ branches. Moreover, even this m_{all} overbound is usually far under PALO$_{all}$'s M_{all}. To motivate this empirical observation, assume each neighborhood has $b = |\mathcal{T}[\Theta_j]|$ elements and that different neighborhoods are effectively disjoint; hence, the total space has something like $|N| = b^K$ elements, where K is the real "depth" of the space. Now observe that PALO$_1$'s k will be under K. In fact, as PALO$_1$ may begin with a non-pessimal element, and also may stop at a non-global optimum, we expect $k \ll K$ in general. Ignoring Equation 8's "$\ln\frac{j^2\pi^2}{3\delta}$" terms and Equation 7's "$\ln\frac{2}{\delta}$", notice that

$$\begin{aligned} m_{all} \;&\approx\; 2\left(\frac{\Lambda_{max}}{\epsilon}\right)^2 \sum_{j=1}^{k}\ln|\mathcal{T}[\Theta_j]| \;=\; 2\left(\frac{\Lambda_{max}}{\epsilon}\right)^2 k\ln b \\ &\le\; 2\left(\frac{\Lambda_{max}}{\epsilon}\right)^2 K\ln b \;\approx\; M_{all} \end{aligned}$$

In fact, $M_{all} > m_{all}$ whenever

$$\ln\frac{2N}{\delta} \;>\; \left(k\ln\frac{b\pi^2k^2}{3\delta}\right) \tag{9}$$

Equation 9 also exposes the infrequent situations where PALO$_1$ can require more samples than PALO$_1$. This can happen when PALO$_1$ starts with an exceptionally bad element, and so is forced to explore essentially the entire space; meaning the "$\ln\frac{\pi^2k^2}{3\delta}$" terms will compensate for the difference between k and K. It can also happen if the neighborhoods have a large overlap, forcing PALO$_1$ to estimate the utilities of the overlapped elements *several* times, which "costs" PALO$_1$ extra samples. By contrast, PALO$_{all}$ will estimate the expected utility

Rule Set

```
R1:  hep(X) :- jaun(X).
R2:  hep(X) :- badB(X).
R3:  badB(X) :- bt#1(X).
R3:  badB(X) :- bt#2(X).
```

Fact Set

```
jaun(b2), jaun(b3), jaun(b5), ...
bt#1(b1), bt#1(b5), bt#1(b7), ...
bt#2(b6), bt#2(b7), ...
```

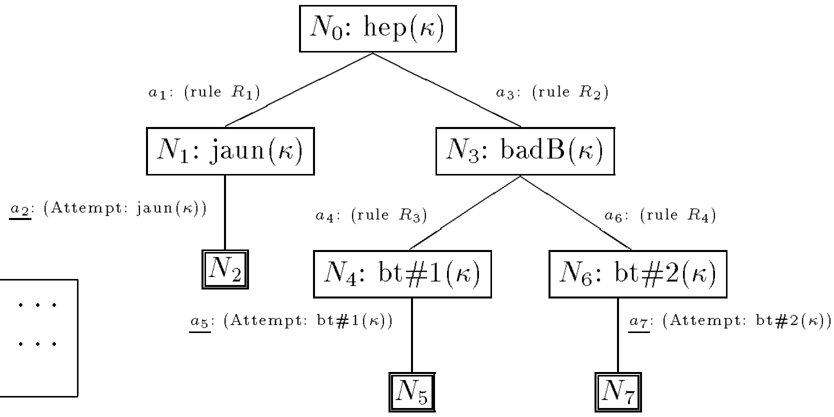

Figure 2: "Inference Graph" G_A, used by Θ_0 and Θ_1

of each element exactly once.

Hence, it is possible that either $m_{all} < M_{all}$ or $m_{all} > M_{all}$, depending on circumstances. However, our experimental data shows that, in practice, PALO$_1$ typically requires *many fewer samples* than PALO$_{all}$, for the basic reasons mentioned above: PALO$_1$ almost always uses well under L_j samples when dealing with the j^{th} element, and PALO$_1$ rarely requires $\ln_b |N|$ iterations before termination.

5 Instantiations of the PALO$_1$ Algorithm

This section demonstrates the generality of the PALO$_1$ algorithm by presenting three different instantiations of this framework. For each instantiation, we specify (1) the set of possible performance elements $\mathcal{S}_\Theta = \{\Theta_j\}$, (2) the set of transformations $\mathcal{T}$ used in the hill-climbing process, and (3) the utility function $c(\cdot, \cdot)$ used to specify the expected utility. We also present the derived values of $\Lambda(\Theta', \Theta)$. (The instantiations of these parameters are summarized in Table 2.) For pedagogical reasons, each subsection begins with a quick simplistic description of its application, and then provides notes that describe how to build a more comprehensive system.

5.1 Improving Efficiency

Many derivation processes can be viewed as a satisficing search [74] through a given graph structure. As an example, notice that using the information shown in Figure 2 to find an answer to the `hep(`κ`)` query, for some ground individual κ, corresponds naturally to a search through the inference graph G_A (formed from the given set of rules) seeking a successful database retrieval.[7] A *strategy* specifies the order in which to perform the various rule-based reductions (*e.g.*, the a_1 arc reduces the "N_0: `hep(`κ`)`" goal to the "N_1: `jaun(`κ`)`" subgoal, based on the rule R_1) and the database retrievals (*e.g.*, the a_2 arc from N_1 to N_2 corresponds to the attempted database retrieval "`jaun(`κ`)`"). We can express each strategy as a sequence of G_A's arcs; *e.g.*, the strategy

$$\Theta_0 \;=\; \langle a_1,\, a_2,\, a_3,\, a_4,\, a_5,\, a_6,\, a_7 \rangle$$

corresponds to the obvious depth-first left-to-right traversal, with the understanding that the performance element using this strategy will stop whenever it has exhausted all of its reductions, or it reaches a "success node" — *e.g.*, if the a_2 retrieval succeeds, then Θ_0 reaches the success node N_2 and so stops with success. (Figure 2 doubly-boxes G_A's success nodes, N_2, N_5 and N_7.) There are many other possible strategies, including various alternative depth-first strategies, such as

$$\begin{aligned} \Theta_1 &= \langle a_1,\, a_2,\, a_3,\, a_6,\, a_7,\, a_4,\, a_5 \rangle \\ \Theta_2 &= \langle a_3,\, a_4,\, a_5,\, a_6,\, a_7,\, a_1,\, a_2 \rangle \\ \Theta_3 &= \langle a_3,\, a_6,\, a_7,\, a_4,\, a_5,\, a_1,\, a_2 \rangle \end{aligned}$$

as well as many non–depth-first strategies.

Each strategy will find an answer, if one exists. As this is a satisficing search, all answers are equally acceptable [74], which means that all strategies are equally *accurate*. We therefore consider the costs of the strategies, preferring the one whose expected cost is minimal.

We use $f_i \in \Re^+$, the non-negative cost of traversing the a_i, to compute $c_s(\Theta,\, q)$, the cost of using strategy Θ to find an answer to the query q. For example, given the atomic propositions in Figure 2's "Fact Set", $c_s(\Theta_0,\, \texttt{hep(b2)}) = f_1{+}f_2$, $c_s(\Theta_0,\, \texttt{hep(b1)}) = f_1{+}f_2{+}f_3{+}f_4{+}f_5$, and $c_s(\Theta_2,\, \texttt{hep(b1)}) = f_3 + f_4 + f_5$. (These different strategies have different costs for a given query as each strategy stops as soon as it has found an answer.) The *expected cost*, of course, depends on the distribution of queries; *i.e.*, on how often the query posed will be `hep(b1)`, versus `hep(b2)`, etc. Moreover, the task of finding the *globally* optimal strategy is NP-hard [30].

This looks like a job for PALO$_1$.[8] We first define the set of reordering transformations $\mathcal{T}^{RO} = \{\tau_{ij}\}$, where each τ_{ij} maps one strategy to another by moving the "subgraph" under the a_i arc to be before a_j and its subgraph. For example, $\tau_{6,4}(\Theta_0) = \langle a_1, a_2, a_3, \boxed{a_6, a_7}, a_4, a_5 \rangle = \Theta_1$ and $\tau_{3,1}(\Theta_0) = \langle \boxed{a_3, a_4, a_5, a_6, a_7}, a_1, a_2 \rangle = \Theta_2$. PALO$_1$ also requires the value of $\Lambda(\Theta, \tau(\Theta))$: these values are bounded by $c(G) = \sum_i f_i$, the sum of the costs of all of the arcs in the inference graph G.

N-EFF1. In general, each class of performance elements is defined with respect to the infer-

[7] Here, `hep(`χ`)` means χ has hepatitis, `jaun(`χ`)` means χ is jaundiced, `badB(`χ`)` means χ has "bad blood", and `bt#i(`χ`)` means χ tests positive for blood test #i. This graph traversal situation corresponds immediately to Section 3's PROLOG situation.

[8] Of course, all of the signs in Figure 1 should be flipped, as we are here measuring *cost* rather than utility, and so prefer the element with *minimal*, rather than maximal, cost. Note also that we are viewing each strategy as a performance element.

ence graph $G = \langle N, A, S, f \rangle$ associated with the given set of rules. In the situation shown above (in which the antecedent of each rule is but a single literal), N is a set of nodes (each corresponding to a proposition; *e.g.*, the node N_0 corresponds to "`hep(`κ`)`" and N_2 corresponds to the empty disjunction), and $A \subset N \times N$ is a set of arcs, each corresponding either a rule-based reduction (*e.g.*, the a_1 arc from N_0 to N_1 is based on the rule R_1) or a database retrieval (*e.g.*, the a_2 arc from N_1 to N_2 corresponds to the attempted database retrieval `jaun(`κ`)`). The set $S \subset N$ is the subset of N's "success nodes" (here, each is an empty disjunction such as N_2 or N_5, shown in doubled boxes); reaching any of these nodes means the proof is successful. The cost function f: $A \mapsto \mathcal{R}_0^+$ maps each arc to a non-negative value that is the cost required to perform this reduction. We earlier let f_i refer to the value of $f(a_i)$.

To deal with more general rules, whose antecedents are conjunctions of more than one literal (*e.g.*, "`a(X) :- b(X), c(X).`"), we must use directed hyper-graphs, where each hyper-arc descends from one node to a *set* of children nodes, where the conjunction of these nodes logically imply their common parent. We must also define S to be a set of subsets of N, where the query processor would have to reach each member of some $s \in S$ for the derivation to succeed. This extension leads to additional complications in specifying strategies; see also [37, Appendix A].

It is also trivial to extend these definitions to accommodate more complicated $f(\cdot)$ cost functions, which can allow the cost of traversing an arc to depend on other factors — *e.g.*, the success or failure of that traversal, which other arcs have already been traversed, etc.

N-EFF2. This class of performance elements corresponds to many standard problem solvers, including PROLOG [12]; see also [24]. We can also use these inference graphs to describe operators working in state spaces; here each internal arc of the inference graph corresponds to an operator invocation and each leaf arc to a general "probabilistic experiment". Using G_A, for example, a_3 encodes the "take some blood" operator, and a_5 encodes the experiment that succeeds if the patient tests positive on `bt#1`, and so forth.

N-EFF3. In [36], we discuss how this instantiation of the PALO$_1$ algorithm fits into the framework of "explanation-based learning" systems, and show in particular, how our framework provides a mathematical basis for [59]'s "utility analysis". We also present a more efficient PALO$_1'$ system that analytically computes upper and lower bounds of $\Delta(\,\Theta,\, \tau_{ij}(\Theta),\, n\,)$ based only on information acquired while running Θ, and then uses this information in Lines $\langle L2 \rangle$ and $\langle L3 \rangle$, respectively, of the code. *N.b.*, this PALO$_1'$ will obtain good estimates of $\Delta(\,\Theta,\, \tau_{ij}(\Theta),\, n\,)$ *without* first constructing $\tau_{ij}(\Theta)$ for each $\tau_{ij} \in \mathcal{T}$, then executing each such element over all $S = \{q_i\}$ queries. That paper also provides a battery of empirical evidence which demonstrate that PALO$_1'$ can work effectively. Jurišica [49] presents an extensive body of related results.

N-EFF4. To illustrate PALO$_1$'s effectiveness, consider again the graph shown in Figure 2, and assume each arc has unit cost — *i.e.*, $f_i = f(a_i) = 1$. We can define the distribution of queries in terms of the (independent) probabilities of the various database retrievals; here,

ϵ	to Θ_2	to Θ_3	to terminate
2.0	22.1		56.5
1.0	25.1	198.6	38.3
0.5	26.7	218.6	82.2
0.1	34.7	276.1	219.7

Table 1: Average number of samples for each PALO$_1$ climb (using Θ_0, $\delta = 0.05$ and various ϵ's)

suppose the real-world distribution is

$$\begin{aligned} Pr[\,\text{jaun}(\kappa) \text{ is in FactSet} \mid \text{query } \texttt{hep}(\kappa) \text{ is posed}\,] &= 0.01 \\ Pr[\,\text{bt\#1}(\kappa) \text{ is in FactSet} \mid \text{query } \texttt{hep}(\kappa) \text{ is posed}\,] &= 0.60 \\ Pr[\,\text{bt\#2}(\kappa) \text{ is in FactSet} \mid \text{query } \texttt{hep}(\kappa) \text{ is posed}\,] &= 0.95 \end{aligned}$$

(Notice these events are not disjoint.) Given the fact set shown, this would happen if `hep(b1)` (resp., `hep(b2)`, `hep(b6)`, `hep(b7)`) was asked 4% (resp., 1%, 39%, 56%) of the time.

We can use these values to compute the expected costs of the various strategies [75]: $C(\Theta_0) = 5.792$, $C(\Theta_1) = 5.069$, $C(\Theta_2) = 3.840$ and $C(\Theta_3) = 3.140$. Of course, as the learner does not initially know these probability values, it will not know that the optimal strategy is Θ_3.

We then ran PALO$_1$ with Θ_0 as the starting element, $\delta = 0.05$, and various settings for ϵ. Using $\epsilon = 2.0$, PALO$_1$ climbed to Θ_2, and usually terminated — which is appropriate, as this Θ_2 is a 2.0-local optimum (even though it is not the global optimum). Over 100 trials, PALO$_1$ required an average of 22.1 samples for the first climb, then about 56.5 additional sample queries to realize this strategy was good enough and terminate; hence, this total learning process required on average about 78.6 total queries.[9] For the smaller values of ϵ, PALO$_1$ always went from Θ_0 to Θ_2 as before, but then used a second hill-climbing step to reach the globally-optimal Θ_3. As expected, the number of steps required for each transition were about the same for all values of $\epsilon = 1.0,\ 0.5,\ 0.1$, requiring on average 25.1, 26.7, 34.7 samples (resp.) to reach Θ_2, then an additional 198.6, 218.6, 276.1 samples to reach Θ_3, and finally 38.3, 82.2, 219.7 more samples to decided that this Θ_3 was in fact an ϵ-local optimum; see Table 1. (Notice the time required to deal with this final set of samples is not wasted: the overall "Θ_3-performance-element–&–PALO$_1$-learning-element" system is still solving relevant, user-supplied, problems, and doing so at a cost that is only slightly more expensive than simply running the Θ_3-performance-element alone, which *we* know is the optimal element.) Figure 3 graphs the $\epsilon = 1.0$ case; the other cases look very similar, of course.

As a final note, the $\delta = 0.05$ setting means that we would allow PALO$_1$ to make 1 mistake in 20 trials. However, in the 400 trials summarized here (involving 100 trials for each for the the 4 values of ϵ), PALO$_1$ *never* made a mistake — *i.e.*, it never climbed to an element that was

[9] In 1 of these 100 trials, PALO$_1$ continued to climb and reached Θ_3; here it required 16 additional samples to terminate.

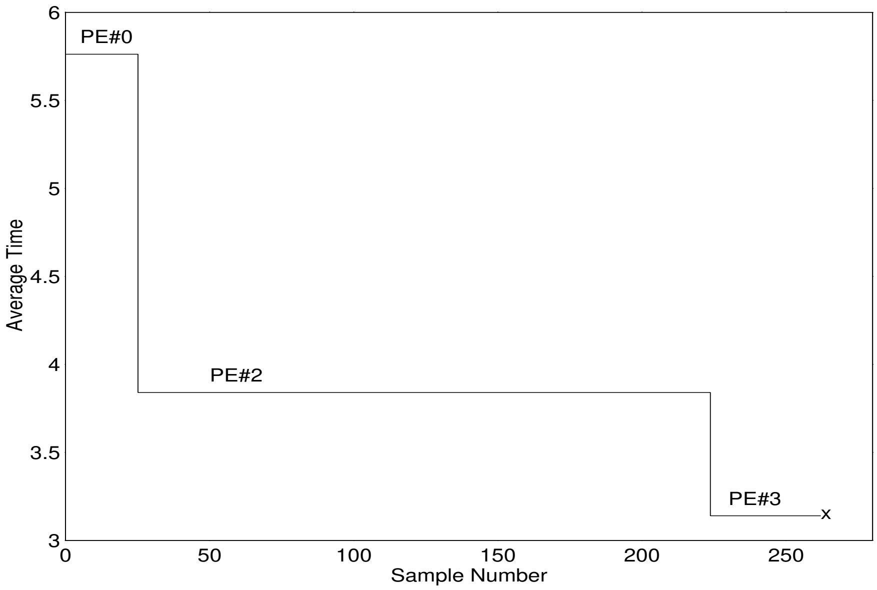

Figure 3: PALO$_1$'s Climbs; Initial element Θ_0, $\epsilon = 1.0$, $\delta = 0.05$

inferior, and it never terminated when there was an ϵ-better neighbor. This, coupled with similar results over several thousand other trials over different inference graphs, and with diverse initial performance elements and values for ϵ and δ, illustrate how overly-conservative PALO$_1$'s statistical tests are.

5.2 Improving Accuracy

A default theory can be ambiguous, as it can produce many individually plausible but collectively incompatible solutions to certain queries [69]. Unfortunately, only (at most) one of these solutions is correct; we would, of course, like to return only that one solution. This is the essence of the "multiple extension" problem in knowledge representation [69, 41, 63], and corresponds to the "bias" and "multiple explanation" problems in machine learning [61, 77, 71, 42] and "reference class" problem in statistics [54, 57]. This subsection addresses this problem by seeking a credulous system, related to the given initial default theory, that is "optimally correct"; *i.e.*, which produces the correct answer most often.

In more detail, we assume there is a correct answer to each query q, denoted $\mathcal{O}(\,q\,)$; hence $\mathcal{O}(\,2+2\;=\;\mathtt{X}\,)\;=\;\mathtt{Yes}[\mathtt{X}\mapsto 4]$. Each correct answer is either "Yes" (possibly with a binding list, as shown here) or "No". Using $\Theta(q)$ to represent the answer returned by the credulous performance element Θ, we can define the utility function

$$c_a(\Theta,\,q) \;\stackrel{def}{=}\; \begin{cases} +1 & \text{if } \Theta(q) = \mathcal{O}(\,q\,) \\ \;\;0 & \text{if } \Theta(q) = \mathtt{IDK} \\ -1 & \text{otherwise} \end{cases} \tag{10}$$

where IDK represents "I don't know".

We focus on stratified THEORIST-style performance elements [66] [67, 9, 79], where each element $\Theta = \langle \mathcal{F},\, \mathcal{H},\, \Upsilon \rangle$ is a triple, composed of a (consistent) set of facts $\mathcal{F}$, a set of allowed hypotheses $\mathcal{H}$ (each a simple type of default [69]) and a specific priority ordering of the hypotheses. As a specific example, consider $\Theta_A = \langle \mathcal{F}_0,\, \mathcal{H}_0,\, \Upsilon_A \rangle$, where[10]

$$\mathcal{F}_0 \quad = \quad \left\{ \begin{array}{l} \texttt{s(X, gray) :- e(X), n}_E\texttt{(X).} \\ \texttt{s(X, white) :- a(X), n}_A\texttt{(X).} \\ \texttt{a(zelda), \ e(zelda), \ ...} \end{array} \right\} \tag{11}$$

is the fact set;

$$\mathcal{H}_0 \quad = \quad \left\{ \begin{array}{ll} h_1\!: & \texttt{n}_E(x) \\ h_2\!: & \texttt{n}_A(x) \end{array} \right\}$$

is the hypothesis set, and $\Upsilon_A = \langle h_1,\, h_2 \rangle$ is the hypothesis ordering.

To explain how Θ_A would process a query, imagine we want to know the color of Zelda — *i.e.*, we want to find a binding for `C` such that σ = "`s(zelda, C)`" holds. Θ_A would first try to prove `s(zelda, C)` from the factual information $\mathcal{F}_0$ alone. This would fail, as we cannot prove that `Zelda` is either a normal elephant or that she is a normal albino (*i.e.*, neither $\texttt{n}_E\texttt{(zelda)}$ nor $\texttt{n}_A\texttt{(zelda)}$ hold, respectively). Θ_A then considers using some hypothesis — *i.e.*, it is allowed to assert an instantiation of some element of $\mathcal{H}_0$ if that proposition is both consistent with the known facts $\mathcal{F}_0$ and if this addition enables us to reach a conclusion to the query posed. Here, Θ_A could consider asserting either $\texttt{n}_E\texttt{(zelda)}$ (meaning that Zelda is a "normal" elephant and hence is colored *gray*) or $\texttt{n}_A\texttt{(zelda)}$ (meaning that Zelda is a "normal" albino and hence is colored *white*). Notice that either of these options, individually, is consistent with everything we know, as encoded by $\mathcal{F}_0$. Unfortunately, we cannot assume both options, as the resulting theory $\mathcal{F}_0 \cup \{\, \texttt{n}_E\texttt{(zelda)},\, \texttt{n}_A\texttt{(zelda)} \,\}$ is inconsistent.

We must, therefore, decide between these options. Θ_A's hypothesis ordering Υ_A specifies the priority of the hypotheses. Here $\Upsilon_A = \langle h_1, h_2 \rangle$ means that $h_1\!:\ \texttt{n}_E(x)$ takes priority over $h_2\!:\ \texttt{n}_A(x)$, which means that Θ_A will return the conclusion associated with $\texttt{n}_E\texttt{(zelda)}$ — *i.e.*, Gray, encoded by $\texttt{Yes[C} \mapsto \texttt{gray]}$, as $\mathcal{F}_0 \cup \{\texttt{n}_E\texttt{(zelda)}\} \models \texttt{s(zelda, gray)}$.[11]

Now consider the $\Theta_B = \langle \mathcal{F}_0, \mathcal{H}_0, \Upsilon_B \rangle$ element, which differs from Θ_A only by including a different priority ordering $\Upsilon_B = \langle h_2, h_1 \rangle$. As Υ_B considers the hypotheses in the opposite order, it will return the answer $\texttt{Yes[C} \mapsto \texttt{white]}$ to this query; *i.e.*, it would claim that Zelda is white.

Which of these two elements is better? If we are only concerned with this single Zelda query, then the better (read "more accurate") Θ_i is the one with the larger value for $c_a(\Theta_i,\, \texttt{s(zelda, C)})$; *i.e.*, the Θ_i for which $\Theta_i(\texttt{s(zelda, C)}) = \mathcal{O}(\,\texttt{s(zelda, C)}\,)$. In general, however, we will have to consider a less trivial distribution of queries. To illustrate this, imagine Equation 11's "..." corresponds to $\{\texttt{a(z}_1\texttt{)}, \texttt{e(z}_1\texttt{)}, \ldots, \texttt{a(z}_{100}\texttt{)}, \texttt{e(z}_{100}\texttt{)}\}$, stating that each $\texttt{z}_i$ is an albino elephant; and that the queries are of the form "$\texttt{s(z}_i\texttt{, C)}$", for various $\texttt{z}_i$s. The best Θ_i now depends on the distribution of queries (*i.e.*, how often

[10] Here `zelda` refers to Zelda, $\texttt{a}(\chi)$ means χ is an albino, $\texttt{e}(\chi)$ means χ is an elephant, and $\texttt{s}(\chi,\ \phi)$ means χ's color is ϕ. The first two clauses in Equation 11 state that normal elephants are gray, and that normal albinos are white. We leave implicit the statments that $\texttt{s}(\cdot,\ \cdot)$ is a function and $\texttt{gray} \neq \texttt{white}$.

[11] This uses the instantiation $\texttt{s(zelda, gray)} = \texttt{s(zelda, C)/Yes[C} \mapsto \texttt{gray]}$. To simplify our notation, we will view "$q/\texttt{No}$" as "$\neg q$".

each "`s(z`$_i$`, C)`" query is posed) and also on the correct answers (*i.e.*, for which $\mathtt{z}_i$s does $\mathcal{O}($ `s(z`$_i$`, C)` $)$ equal `Yes[C` $\mapsto$ `white]` as opposed to `Yes[C` $\mapsto$ `gray]`, or some other answer). Hence, the expected accuracy of each system $\mathrm{C}_a(\, \Theta_i\,)$, is defined by plugging Equation 10's $c_a(\cdot, \cdot)$ function into Equation 2; we would then select the Θ_i system with the larger $\mathrm{C}_a(\, \cdot\,)$ value.

In general, $\Theta = \langle \mathcal{F}, \mathcal{H}, \Upsilon \rangle$ can include a much larger set of hypotheses $\mathcal{H} = \{h_1, \ldots h_n\}$. As before, each ordering $\Upsilon = \langle h_{\Gamma(1)}, \ldots h_{\Gamma(n)} \rangle$ is a sequence of $\mathcal{H}$'s elements, based on the permutation $\Gamma\colon [1..N] \mapsto [1..N]$. Θ uses this information when answering queries: Let i be the smallest index such that $\mathcal{F} \cup \{h_{\Gamma(i)}\}$ is consistent and $\mathcal{F} \cup \{h_{\Gamma(i)}\} \models q/\beta_i$ for some answer β_i; here Θ returns this β_i.[12] If there is no such i, then Θ returns `IDK`.

Our goal is to identify the priority ordering that is accurate most often. As before, this depends on the distribution of queries, which unfortunately is not known *a priori*; and moreover, the task of identifying this optimal ordering of the hypotheses is NP-complete (and worse, not even approximatable [1]), even if we knew the distribution, even in the simplistic situation that we have been considering, where every derivation involves exactly one hypothesis, etc. [32].

Once again, PALO$_1$ is designed to deal with this situation. We first define the set of transformations $\mathcal{T}^A = \{\, \tau_{ij}\, \}_{i,j}$, where each τ_{ij} moves the j^{th} term in the ordering to just before the i^{th} term — *i.e.*, given any ordering $\Upsilon = \langle h_1, h_2, \ldots, h_n \rangle$,

$$\tau_{ij}(\Upsilon) = \langle h_1, \ldots, h_{i-1}, \underline{h_j}, h_i, \ldots, h_{j-1},\ h_{j+1}, \ldots, h_n \rangle\ .$$

We can compute the value of $\Delta(\, \Upsilon_k,\, \tau_{ij}(\Upsilon_k),\, n\,)$ for each τ_{ij} transformation and each set of queries $\{q_m\}_{m=1}^n$ based on whether $\mathcal{F} \cup \{h_\ell\} \models^? q_m/\mathcal{O}(\, q_m\,)$ for each hypothesis h_ℓ. Observe finally that $\Lambda(\Theta, \tau(\Theta)) \leq 2$ for all $\Theta \in \mathcal{S}_\Theta$ and all $\tau \in \mathcal{T}^A$.

N-ACC1. The motivation underlying this work is similar to the research of Shastri [73] and others, who also use probabilistic information to find an ordering of the given default rules. Our work differs by providing a way of obtaining the relevant statistics, rather than assuming that they are known *a priori*, or can be computed purely from static analysis of ground facts in the database.

N-ACC2. In many situations, we may want to consider each hypothesis to be the conjunction of a *set* of sub-hypotheses, which must all collectively be asserted to reach a conclusion. Here, we can view $\mathcal{H} = \mathcal{P}[H]$ as the power set of some set of sub-hypotheses H.

N-ACC3. The description so far assumes that every ordering of hypotheses is meaningful. In some contexts, there may already be a meaningful partial ordering of the hypotheses, perhaps based on specificity or some other criteria [40]. Here, we can still use PALO$_1$ to complete the partial ordering, by determining the relative priorities of the initially incomparable elements.

[12] Technically, our performance element considers adding in some *instantiation* of $h_{\Gamma(i)}$ (*i.e.*, $h_{\Gamma(i)}/\phi$ for some binding list ϕ), and so we are seeking the smallest index i such that $\mathcal{F} \cup \{h_{\Gamma(i)}/\phi\}$ is consistent and $\mathcal{F} \cup \{h_{\Gamma(i)}/\phi\} \models q/\beta_i$ for some binding lists ϕ, β_i.

N-ACC4. As this PALO$_1$ process can require general theorem proving (*e.g.*, to determine whether $\mathcal{F} \cup \{h_i\} \models^? q/\mathcal{O}(q)$), it can be undecidable in general. We of course need to insist that this process be decidable to guarantee that PALO$_1$ will terminate with probability 1. We can, however, guarantee that each of PALO$_1$'s iterations will be polytime if the $\mathcal{F} \cup \{h_i\} \models^? q_k$ computation is polytime (*e.g.*, if we are dealing with propositional Horn theories or propositional 2-CNF, etc. [8]).

N-ACC5. Recall that, in general, we need to compute the values of $c(\tau_{ij}(\Upsilon_\ell), q) - c(\Upsilon_\ell, q)$ for each τ_{ij} in $\mathcal{T}^A$. Above, we obtained this information by determining whether $\mathcal{F} \cup \{h_i\} \models^? q/\mathcal{O}(q)$ holds for each hypothesis h_i. In some situations, there can be more efficient ways of estimating these values, perhaps by using some Horn approximation to $\mathcal{F} \cup \{h_i\}$; see Section 5.3 below. We can also simplify the computation if the $\{h_j\}$ hypotheses are not independent; *e.g.*, if each corresponds to a set of sub-hypotheses, as discussed in Note N-ACC2 above.

N-ACC6. This paper considers only one type of transformation to convert one theory into another — *viz.*, by rearranging the set of hypotheses. There are many other approaches, *e.g.*, by eliminating some inappropriate sets of hypotheses [13, 82], or by modifying the antecedents of individual rules [65], etc. Each of these approaches can be viewed as using a set of transformations to navigate around a space of interrelated theories. We can then consider the same objective described above: to identify the element in the implied space that has the highest expected accuracy.

Here, as above, the expected accuracy score for each element depends on the unknown distribution, meaning we will need to use some sampling process. In some simple cases, we may be able to identify (an approximation to) the *globally* optimal element with high probability; *cf.*, the PAO algorithm discussed in [38]. In almost all cases, however, this identification task is intractable, and not even approximatable [33]. Here again it makes sense to use a hill-climbing system like PALO$_1$ to identify an element that is close to a local optimum, with high probability. (Of course, this local optimality will be based on the classes of transformations used to define the space of theories, etc.)

N-ACC7. There are several obvious extensions to this task: First, our model assumes that each answer to a query is either completely correct or completely false; in general, we can imagine a range of answers to a query, some of which are better than others. (For example, the correct answer to a particular existential query could be a set of 10 distinct instantiations. Here, returning 9 of them may be better than returning 0, or than returning 1 wrong answer. As another situation, we may be able to rank responses in terms of their precision: *e.g.*, knowing that the cost of `watch`$_7$ is `$2,000` is more precise than knowing only that `watch`$_7$ is `expensive` [81].) We have also assumed that all queries are equally important; *i.e.*, a wrong answer to any query "costs" us the same -1, whether we are asking for the location of a salt-shaker, or of a stalking tiger. One way of addressing all of these points is to permit the user to specify her own general $c(\Theta, q)$ function, which could incorporate these different factors, by differentially weighting the different queries, the different possible answers, etc.

On a related theme, notice that this subsection has completely ignored the computational cost of obtaining that answer. Within our framework, however, we can consider yet more

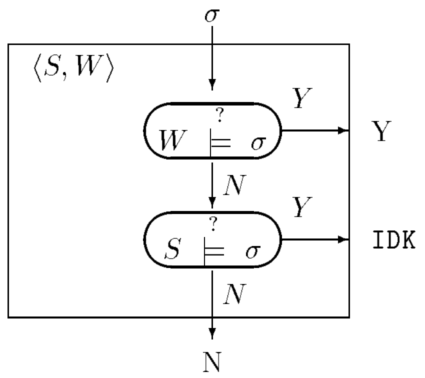

Figure 4: Flow Diagram of $\langle S, W\rangle$ addressing $\Sigma \models^? \sigma$

general $c(\cdot, \cdot)$ functions, that can incorporate the user's tradeoffs between accuracy and efficiency, etc. This would allow the user to prefer, for example, a performance system that returns `IDK` in complex situations, rather than spend a long time returning the correct answer; or even allow it to be wrong in some instances [34].

5.3 Improving Categoricity

The task of determining whether a query is entailed by a theory is known to be intractable if the theory is a general propositional theory (assuming $P \neq NP$) [14, 23]. It can, however, be performed efficiently if the theory contains only Horn clauses [18].[13] Selman and Kautz [72] use this observation to define a particular "knowledge compilation" method: Given a general propositional theory Σ, their compiler computes a pair of "bracketing" Horn theories S and W, with the property $S \models \Sigma \models W$; we call each such S a "$\underline{S}$trengthening" of the initial theory Σ, and each such W a "$\underline{W}$eakening". Figure 4 shows how the resulting "compiled system" $\Theta = \langle S, W\rangle$ uses these bracketing theories to determine whether a query σ follows from Σ: If $W \models \sigma$, then Θ terminates with "yes"; otherwise, if $S \not\models \sigma$, then Θ terminates with "no". (Notice that these are the correct answers, in that $W \models \sigma$ guarantees that $\Sigma \models \sigma$, and $S \not\models \sigma$ guarantees that $\Sigma \not\models \sigma$. Moreover, these tests are linear in the sizes of σ and S (respectively, σ and W), provided $\neg\sigma$ is Horn [18].[14]) Otherwise, if $W \not\models \sigma$ and $S \models \sigma$, Θ returns `IDK`. Notice this compiled system is usually tractable,[15] yet can deal with an arbitrary propositional theory. However, it may not be completely categoric, as it may return `IDK` for some queries whose answers should be either `Yes` or `No`. Hence, we have sacrificed completeness for tractability.

[13] A clausal theory is 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. When convenient, we will write each clause as either a disjunction of literals, or as a set of literals; *e.g.*, $\gamma \equiv a_1 \vee a_2 \vee \neg a_3$ is equivalent to $\gamma = \{a_1, a_2, \neg a_3\}$.

[14] We can actually allow the query σ to be a conjunction of "Horn-dual" propositions, where a proposition σ is a Horn-dual iff its negation $\neg\sigma$ is Horn. Notice this class of Horn-duals strictly includes CNF.

[15] Note N-Cat1 below explains this caveat.

We of course would like to use an approximation $\langle S_i, W_i \rangle$ that is as categorical as possible; *i.e.*, which minimizes the probability that the associated $\langle S_i, W_i \rangle$ system will return `IDK`. To state this more precisely: Given any approximation $\langle S,\, W \rangle$ and query σ, let $c_c(\langle S,\, W \rangle,\, \sigma) \stackrel{def}{=} d(W,\, \sigma) + (1 - d(S,\, \sigma))$ where

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

Hence, $c_c(\langle S,\, W \rangle,\, \sigma) = 1$ if σ is "covered" by $\langle S,\, W \rangle$, in that either $W \models \sigma$ or $S \not\models \sigma$. Using Equation 2, we can then define $\mathrm{C}_c(\,\langle S,\, W \rangle\,)$ to be the expected value of $c_c(\langle S,\, W \rangle,\, \cdot)$. Our goal is to determine the approximation $\langle S,\, W \rangle$ with the largest $\mathrm{C}_c(\,\cdot\,)$ value. As before, this task is intractable (see [72]) and depends on the distribution, suggesting yet again that we use the PALO_1 system.

Observe that the set of queries covered by a strengthening and a weakening are disjoint — *i.e.*, for any approximation $\langle S, W \rangle$, there is no query σ such that both $W \models \sigma$ and $S \not\models \sigma$. This means an approximation $\langle S_i, W_j \rangle$ is, with probability at least $1 - \delta$, within ϵ of a local optimum if S_i (resp., W_j) is within ϵ of a locally optimal strengthening (resp., weakening) with probability at least $1 - \delta/2$. We can therefore decouple the task of finding a good strengthening from that of finding a good weakening, and handle each separately. This paper focuses on how to find a good strengthening; Note N-Cat1 below discusses how to compute a good weakening.

Hence, we are seeking a strengthening S_{opt} whose $\mathrm{D}(\,S_{opt}\,)$ value is minimal, where $\mathrm{D}(\,S_{opt}\,) \;=\; E[\,d(S,\,\cdot)\,]$ is the expected value of $d(S,\,\cdot)$. (Recall we want $S_{opt} \models \sigma$ to *fail* for as many queries as possible.) It is easy to see that this S_{opt} should be a weakest strengthening; *i.e.*, satisfy $\mathrm{OptS}(\Sigma, S_{opt})$ where

$$\begin{array}{rcl} \mathrm{OptS}(\Sigma, S) & \iff & S \models \Sigma \;\&\; \mathrm{Horn}(S) \;\& \\ & & [\neg\exists\, T.\; [S \models T \models \Sigma \;\&\; \mathrm{Horn}(T) \;\&\; S \not\equiv T] \end{array}$$

To compute these OptSs: Define a "horn-strengthening" of the clause $\gamma = \{a_1,\, \ldots,\, a_k,\, \neg b_1,\, \ldots,\, \neg b_\ell\}$ to be 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 γ's positive literals. Here, there are k horn-strengthenings of this γ, each of the form $\gamma_j = \{a_j,\ \neg b_1, \ldots, \neg b_\ell\}$ for some $j = 1..k$. For example, the two horn-strengthening of the non-Horn clause $\gamma \equiv a_1 \vee a_2 \vee \neg b_1 \vee \neg b_2$ are $\gamma_1 \equiv a_1 \vee \neg b_1 \vee \neg b_2$ and $\gamma_2 \equiv a_2 \vee \neg b_1 \vee \neg b_2$.

Now write $\Sigma = \Sigma_H \cup \Sigma_N$, where Σ_H is the subset of Σ's clauses that are Horn and $\Sigma_N = \{\gamma^i\}_{i=1}^m$ is its non-Horn subset. Selman and Kautz [72] prove that each optimal strengthening is of the form $S_o = \Sigma_H \cup \Sigma'_N$, where each $\gamma' \in \Sigma'_N$ is a horn-strengthening of some $\gamma \in \Sigma_N$. By identifying each Horn-strengthened theory with the "index" of the positive literal used (*i.e.*, using γ_j of the form shown above, ${\gamma^i}_j = \{a^i_j,\ \neg b^i_1, \ldots, \neg b^i_{\ell_i}\}$), we can consider any Horn-strengthened theory to be a set of the form $S_{\langle j(1), j(2), \ldots, j(m) \rangle} = \Sigma_H \cup \{\gamma^1_{j(1)}, \gamma^2_{j(2)}, \ldots, \gamma^m_{j(m)}\}$. For example, given the theory $\Sigma = \Sigma_N \cup \Sigma_H$ with non-Horn clauses $\Sigma_N = \{\gamma_1, \gamma_2\}$ where

$\gamma_1 = a_1 \vee a_2 \vee \neg b_1 \vee \neg b_2$ and $\gamma_2 = c_1 \vee c_2 \vee c_3 \vee \neg d_1$, the $\langle 1,\, 3\rangle$ strengthening would be

$$S_{\langle 1,3\rangle} \quad = \quad \Sigma_H \;\cup \left\{ \begin{array}{lll} a_1 & & \vee \neg b_1 \;\vee\; \neg b_2 \\ & c_3 & \vee \neg d_1 \end{array} \right\}$$

We can navigate about this space of Horn-strengthened theories by changing the index associated with individual non-Horn clauses: That is, define the set of transformations $\mathcal{T}^S = \{\tau_{k,\ell}\}_{1\leq k\leq m; 1\leq \ell \leq n}$ where each $\tau_{k,\ell}$ is a function that maps one strengthening to another by changing the "index" of the k^{th} clause to be ℓ; *e.g.*, $\tau_{k,\ell}(S_{\langle 3,9,\ldots,i_k,\ldots,5\rangle}) = S_{\langle 3,9,\ldots,\ell,\ldots,5\rangle}$. (Of course, m is the number of non-Horn clauses in Σ and n is the total number of propositional variables in the theory.) Continuing with the earlier example,

$$\tau_{2,1}(S_{\langle 1,3\rangle}) \;=\; S_{\langle 1,1\rangle} \;=\; \Sigma_H \;\cup \left\{ \begin{array}{ll} a_1 & \vee \neg b_1 \;\vee\; \neg b_2 \\ c_1 & \vee \neg d_1 \end{array} \right\}$$

and

$$\tau_{1,2}(S_{\langle 1,3\rangle}) \;=\; S_{\langle 2,3\rangle} \;=\; \Sigma_H \;\cup \left\{ \begin{array}{lll} a_2 & & \vee \neg b_1 \;\vee\; \neg b_2 \\ & c_3 & \vee \neg d_1 \end{array} \right\}$$

This instantiation of the PALO_1 process starts with any given Horn-strengthened theory (perhaps $S_{\langle 1,1,\ldots,1\rangle}$) and hill-climbs in the space of Horn-strengthened theories, using this set of $\mathcal{T}^S$ transformations. As $\Delta(\, S_i,\, S',\, n\,)$ depends only on whether $S' \models \sigma$ and $S_i \models \sigma$, it can be answered efficiently, as each S_i and S' are Horn. (In fact, this process can also use the support of σ from S_i to further improve its efficiency.) Notice finally that $\Lambda(S_i, S') \leq 1$ for all strengthenings S_i, S'.

N-CAT1. Selman and Kautz [72] prove that there is a unique optimal weakening, w_s, which corresponds to the set of all horn implicates of the initial theory. Unfortunately, this w_s can be exponentially larger than the original theory [51]. This means the cost of using $\langle S, w_s\rangle$ to answer queries can be exponential, as the complexity of $w_s \models \sigma$ is linear in $|w_s| = O(2^{|\Sigma|})$. (This was not an issue with strengthenings, as each strengthening S_i is "small", in fact, $|S_i| \leq |\Sigma|$, where the size of a horn theory $|T|$ is the number of clauses in T.)

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, designed to implement the user's tradeoffs between efficiency and categoricity. Our goal, therefore, is to find the weakening *of this size* that is maximally categorical, over the distribution of queries. Once again, this best K-sized weakening depends on the distribution, and moreover, the task of determining which is best, even given the distribution, is again intractable; see Theorem 2 in the appendix.

This motivates us to use PALO_1 for this subtask as well. The space of transformations is more complicated to describe, however. [72] provides an (exponential time) algorithm LUB for computing the optimal weakening w_s. In essence, this algorithm iteratively attempts to resolve each horn clause with each non-horn clause, and adds in each successful resolvent, after removing all subsumed clauses. Each of our transformations $\tau_k \in \mathcal{T}^W$ performs one such step. There is, of course, the additional challenge of keeping the total number of clauses bounded; this forces the transformations to remove a clause from the current approximation, to make room for each proposed addition.

We discuss these transformations in detail in [39], and also present a PALO_1-ish algorithm, called AdComp, that hill-climbs in both spaces, to find both a near-optimal strengthening and a near-optimal weakening. This algorithm is tractable in two senses: The $\langle S_i, W_j \rangle$ approximation it produces admits efficient computation, as both S_i and W_j are Horn and of bounded size; and also each of AdComp's steps is guaranteed to be tractable, basically because AdComp never computes $\Sigma \models \sigma$. Instead, AdComp approximates this computation using only the current theories S_i and W_j, together with the neighbors $\tau(S_i)$ and $\tau(W_j)$. As there are only a polynomial number of neighbors, and each theory is both Horn and of bounded size, the overall computation that approximates $\Sigma \models \sigma$ is efficient.

N-Cat2. The user-specified $K(\cdot)$ function, used to bound the size of the weakening, implicitly quantifies how much time the user will allow the system to spend trying to answer a query before insisting that it stop and return `IDK`. We can generalize this idea by allowing the user to specify her own general utility measure $c_i(\cdot,\,\cdot)$, where $c_i(\langle S,\, W\rangle,\, \sigma)$ quantifies how well the approximation $\langle S,\, W\rangle$ does at solving σ, which can specify an arbitrary combination of various factors, including accuracy, categoricity and efficiency. (See also the discussion in Note N-Acc7 above.)

On a related theme:, our current $\langle S_i, W_j \rangle$ system returns `IDK` if $W \not\models \sigma$ and $S \models \sigma$. There are many other options for this situation — *e.g.*, perhaps Θ should "guess" at an answer here, or alternatively spend as long as necessary to compute whether $\Sigma \models^? \sigma$, etc. Of course, this depends on the user's objectives, which can be incorporated within her $c_i(\langle S,\, W\rangle,\, \sigma)$ utility function; we could then use a PALO-like system to climb in the space of such performance elements (which differ in their way of handling the "$W \not\models \sigma$ and $S \models \sigma$" situation), in parallel with its search for good weakenings and strengthenings; see [39].

N-Cat3. This compilation work is obviously related to the work on "vividization" and "approximation" [56, 7, 20, 15, 47], which also try to transform a given intractable theory into a representation that admits more efficient, if less categorical, reasoning. Our work extends those results by (1) quantifying the goal of producing a system that is maximally categorical *over the anticipated distribution of queries*; and (2) by providing an efficient, autonomous way of computing such an efficient approximation.

6 Conclusion

6.1 Other Variants of PALO Systems

As suggested by the "1" subscript of "PALO_1", there is a family of algorithms that each satisfy the properties specified in Theorem 1. The technical note [31] presents various other PALO_i algorithms, which differ from PALO_1 in small, but significant, ways. While PALO_1 consider climbing or terminating *after each sample*, the simpler "on-line but batched" PALO_0 system instead examines an entire collection of L'_j samples at a time. (Unlike the PALO_{all} of Note N-Palo5, this PALO_0 uses different samples for different climbs.) Another alternative is the PALO_2 system, which is guaranteed to climb only a bounded number of times, where the bound can be precisely specified. Finally, the PALO_{1N} system makes stronger assumptions about the world; *viz.*, that the distribution from which the samples are drawn is *normal*. This allows it to use to use lower "barriers" before deciding to climb, or to terminate. Of course, the PALO$_{1N}$ system is more likely to make mistakes if the data is not really drawn from a normal distribution. (There are also PALO$_{0N}$ and PALO$_{2N}$ algorithms, which differ from PALO$_0$ and PALO$_2$ only by making normality assumptions. Notice also that these splits, of PALO$_0$ versus PALO$_1$ versus PALO$_2$, and PALO$_\chi$ versus PALO$_{\chi N}$, are orthogonal to one another, and also orthogonal to the set of the applications discussed in Section 5 above. In particular, any of these PALO$_i$ systems can be used to address any of these applications.)

We [31] empirically tested these different PALO$_i$ systems in several different contexts, and found that the PALO$_1$ system discussed here was usually the best, in terms of the utility of its final performance element, as a function of the empirical sample complexity. More recently, however, we found that PALO$_{1N}$ worked effectively in one particular context; see [35].

6.2 Limitations

The examples discussed in Section 5 illustrate the versatility and generality of the PALO objective, of identifying a performance element whose expected utility, over an arbitrary (but stationary) distribution of problems, is optimal. Our particular PALO system is designed to handle the situations when (1) the distribution of samples, which is required to determine the expected values, is not known *a priori*, and (2) the task of computing the optimal element based on this quality measure, once given the distribution information, is intractable. The situations presented above illustrate that this is a very common situation.

It is worth discussing PALO's limitations as well, to understand when it should *not* be applied. PALO is best used when the goal is to find a performance element whose *expected* utility is maximal. Notice first that this implies that the distribution is stationary; if not, then even defining this *expected* utility can be problematic. Second, PALO is not useful if the task is inherently *mini-max*: seeking the element whose worst case performance is as good as possible. For example, imagine we need a system that will *always* return an answer within say 1 second; *i.e.*, there is no additional benefit for taking under 1 second, but an extreme penalty for taking any longer than 1 second. Here, it is critical to know the worst possible time a performance element can require. To illustrate this, imagine Θ_1 requires 0.5-seconds for *all* samples, while Θ_2 requires only 0.001 seconds for all but one extremely rare query q_{bad}, which appears only 0.001% of the time, but Θ_2 requires 2 seconds for this q_{bad}. Here, even though Θ_2 clearly has a better *expected* efficiency than Θ_1, its worst-case efficiency is worse; which means, by the above criterion, that Θ_1 should is prefered. (Of course, we could model this situation by defining $c(\Theta,\, q) = 0$ iff Θ takes under 1 second for q, and $-\infty$ otherwise. Here, however, the value of Equation 1's $\lambda = \lambda(c, \mathcal{Q}, \mathcal{S}_\Theta)$ would be infinite, which would prevent PALO from ever either climbing or terminating.)

Another limitation is that PALO may not reach the global optimum, as it is only hill-climbing. Worse, it may not even find a local optimum, as it is only (probabilistically) guaranteed to find an ϵ-local optimum, which means it will stop on reaching a "gentle slope" — *i.e.*, when Θ's neighbors are, at best, only slightly (read "under ϵ") better than Θ. As mentioned earlier, we could build a PALO-like system that attempts to avoids such non-global ϵ-local optima by stochastically descending, *á la* simulated annealing. This approach,

	Efficiency	Accuracy	Categoricity
Performance Elements $\mathcal{S}_\Theta$	satisficing strategies	hypothesis orderings	Horn-strengthenings
Utility function $c(\cdot,\cdot)$	computation time	$\Theta(q) =^? \mathcal{O}(q)$	$S \models^? q \quad (W \not\models^? q)$
Transformations $\mathcal{T}$	reorder arcs	reorder priority	change 1 clause
Range $\Lambda[\Theta, \tau(\Theta)]$	$\leq c(G)$	≤ 2	≤ 1

Table 2: Summary of Applications

however, sacrifices the guarantees proven above.

The conditions above specify situations where PALO will not work effectively. There are also situations where PALO *should not* be used. For example, there is no need to use the weak hill-climbing method if there is a known efficient techniques for computing the global optimum; *cf.*, [78, 2]. Here, it is often sufficient to simply estimate the distribution, then let the efficient algorithm use that estimate. PALO can also be inappropriate if the distribution is known initially; here, it is probably better to simply run a standard hill-climbing algorithm, using as the quality measure for each element the actual expected utility, computed directly rather than estimated. Similarly, there are obvious variants of PALO that can be used when the distribution of the samples is constrained, perhaps by being known to be Gaussian, etc.; these variants may be more sample efficient. (See the PALO_{1N} system mentioned above.)

6.3 Contributions

This paper first poses two of the problems that can arise in learning systems that seek a performance element whose expected utility is optimal [43, 80]: *viz.*, that the distribution information (which is required to determine which element is optimal) is usually unknown, and that finding a globally optimal performance element can be intractable. It then presents an algorithm, PALO_1, that side-steps these shortcomings by using a statistical technique to approximate the distribution, and by hill-climbing to produce a locally optimal element. After defining this algorithm and specifying its behavior, we demonstrate PALO_1's generality by showing that it can be used to find a near-optimal element in three different settings, based on different spaces of performance elements and different criteria for optimality: efficiency, accuracy and categoricity. (See Table 2.) These results suggest approaches to solving (respectively) the utility problem from explanation-based learning, the multiple extension problem from nonmonotonic reasoning and the tractability/completeness tradeoff problem from knowledge representation.

A Proofs

Theorem 1 *The* $\text{PALO}_1(\Theta_1, \mathcal{T}, \epsilon, \delta)$ *process incrementally produces a series of performance elements* $\Theta_1, \Theta_2, \ldots, \Theta_m$, *such that each* $\Theta_{j+1} = \tau_{kj}(\Theta_j)$ *for some* $\tau_{k_j} \in \mathcal{T}$ *and, with probability at least* $1 - \delta$,

1. *the expected utility of each performance element is strictly better than its predecessors i.e.,* $\forall 1 \leq i < j \leq m: \; C(\Theta_j) \; > \; C(\Theta_i)$; *and*

2. *the final performance element returned by* PALO$_1$, Θ_m, *is an "ϵ-local optimum" — i.e.,* $\neg\exists \tau \in \mathcal{T}: \quad C(\,\tau(\Theta_m)\,) \; \geq \; C(\,\Theta_m\,) + \epsilon$.

Moreover, PALO$_1$ *will stay at any* Θ_j *(before either terminating or climbing to a new* Θ_{j+1}*) for a number of samples that is polynomial in* $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $\lambda(c, \mathcal{Q}, \mathcal{S}_\Theta)$ *and* $|\mathcal{T}[\Theta_j]|$, *and will terminate with probability* 1, *provided* $|\mathcal{S}_\Theta|$ *is finite.*

Proof: To prove Parts 1 and 2, consider first a single stage of the PALO$_1$ algorithm, when it is dealing with Θ_j. Notice there are four types of mistakes that PALO$_1$ could make:

A. After seeing i samples (for some $i = 1..L_j - 1$), PALO$_1$ could climb from Θ_j to some $\Theta' = \tau(\Theta_j)$ as Θ' appears to be better than Θ_j (based on Line $\langle L3 \rangle$ test), but in reality, Θ' is not better; or

B. After seeing i samples (for some $i = 1..L_j - 1$), PALO$_1$ could terminate as no $\Theta' = \tau(\Theta_j)$ appears to be more than ϵ better than Θ_j (based on Line $\langle L4 \rangle$ test), but there is some Θ' that is much better; or

C. After seeing all L_j samples, PALO$_1$ could climb from Θ_j to some $\Theta' = \tau(\Theta_j)$ as Θ' appears to be better than Θ_j (based on Line $\langle L5 \rangle$ test), but in reality, Θ' is not better; or

D. After seeing all L_j samples, PALO$_1$ could terminate as no $\Theta' = \tau(\Theta_j)$ appears to be more than ϵ better than Θ_j, but there is some Θ' that is much better.

Using $\epsilon_i(\Theta') = \Lambda(\Theta_j, \Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2\,(L_j - 1)\,|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}$ as the "barrier" used to decide whether to climb to the neighboring $\Theta' = \tau(\Theta_j)$ after seeing i samples, the respective probabilities of these events are

$$
\begin{aligned}
a_j^i &= Pr\Big[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]:\; \tfrac{1}{i}\Delta(\,\Theta_j,\,\Theta',\,i\,) \geq \epsilon_i(\Theta') \text{ and } \mathrm{C}(\,\Theta'\,) \,<\, \mathrm{C}(\,\Theta_j\,)\,\Big] \\
b_j^i &= Pr\Big[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]:\; \tfrac{1}{i}\Delta(\,\Theta_j,\,\Theta',\,i\,) \,<\, \epsilon - \epsilon_i(\Theta') \text{ and } \mathrm{C}(\,\Theta'\,) \,>\, \mathrm{C}(\,\Theta_j\,) + \epsilon\,\Big] \\
c_j &= Pr\Big[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]:\; \tfrac{1}{L_j}\Delta(\,\Theta_j,\,\Theta',\,L_j\,) \,\geq\, \tfrac{\epsilon}{2} \text{ and } \mathrm{C}(\,\Theta'\,) \,<\, \mathrm{C}(\,\Theta_j\,)\,\Big] \\
d_j &= Pr\Big[\,\exists\,\Theta' \in \mathcal{T}[\Theta_j]:\; \tfrac{1}{L_j}\Delta(\,\Theta_j,\,\Theta',\,L_j\,) \,<\, \tfrac{\epsilon}{2} \text{ and } \mathrm{C}(\,\Theta_j\,) \,>\, \mathrm{C}(\,\Theta_j\,) + \epsilon\,\Big]
\end{aligned}
$$

Of course, each existential within a $Pr[\cdot]$ above is really a finite disjunction, ranging over elements of $\mathcal{T}[\Theta_j]$. Moreover, a_j^i is only considering the subset of Θ's for which $\mathrm{C}(\,\Theta'\,) < \mathrm{C}(\,\Theta_j\,)$; hence using $\mathcal{T}^< \; = \; \{\,\Theta' \in \mathcal{T}[\Theta_j] \,|\, \mathrm{C}(\,\Theta'\,) < \mathrm{C}(\,\Theta_j\,)\,\}$,

$$
a_j^i = Pr\left[\bigvee_{\Theta' \in \mathcal{T}^<} \frac{1}{i}\Delta(\,\Theta_j,\,\Theta',\,i\,) \,\geq\, \epsilon_i(\Theta')\right]
$$

and using $\mathcal{T}^{>} = \{\Theta' \in \mathcal{T}[\Theta_j] \mid \mathrm{C}(\Theta') > \mathrm{C}(\Theta_j) + \epsilon\}$,

$$b_j^i = Pr\left[\bigvee_{\Theta' \in \mathcal{T}^{>}} \frac{1}{i}\Delta(\Theta_j, \Theta', i) \leq \epsilon - \epsilon_i(\Theta')\right]$$

and so on.

Now observe that

$$\begin{aligned}
a_j^i &\leq \sum_{\Theta' \in \mathcal{T}^{<}} Pr\left[\frac{1}{i}\Delta(\Theta_j, \Theta', i) \geq \epsilon_i(\Theta')\right] \\
&\leq \sum_{\Theta' \in \mathcal{T}^{<}} Pr\left[\frac{1}{i}\Delta(\Theta_j, \Theta', i) \geq (\mathrm{C}(\Theta') - \mathrm{C}(\Theta_j)) + \epsilon_i(\Theta')\right] &\quad (12)\\
&\leq \sum_{\Theta' \in \mathcal{T}^{<}} e^{-2i\left(\frac{\epsilon_i(\Theta')}{\Lambda(\Theta_j,\Theta')}\right)^2} &\quad (13)\\
&\leq |\mathcal{T}^{<}|\; exp\left\{-2i\left(\frac{\Lambda(\Theta_j,\Theta')\sqrt{\frac{1}{2i}\ln\left(\frac{2(L_j-1)|\mathcal{T}[\Theta_j]|}{\delta_j}\right)}}{\Lambda(\Theta_j,\Theta')}\right)^2\right\}\\
&= |\mathcal{T}^{<}| \frac{\delta_j}{2(L_j-1)|\mathcal{T}[\Theta_j]|}
\end{aligned}$$

Line 12 follows from the observation that $\frac{1}{i}\Delta(\Theta_j, \Theta', i) \geq \epsilon_i(\Theta')$ and $\mathrm{C}(\tau(\Theta_j)) - \mathrm{C}(\Theta_j) < 0$ implies $\frac{1}{i}\Delta(\Theta_j, \tau(\Theta_j), i) \geq (\mathrm{C}(\tau(\Theta_j)) - \mathrm{C}(\Theta_j)) + \epsilon_i(\Theta')$ (using $A \Rightarrow B$ implies $Pr[A] \leq Pr[B]$, for any events A and B). Line 13 uses Hoeffding's inequality (Equation 6) based on the realization that $\frac{1}{i}\Delta(\Theta_j, \tau(\Theta_j), i)$ is the empirical average of a set of i independent, identically-distributed random values $\{c(\tau(\Theta_j), q_\ell) - c(\Theta_j, q_\ell)\}_{\ell=1}^{i}$, whose (common) mean is $\mathrm{C}(\Theta') - \mathrm{C}(\Theta_j)$, and whose range of possible values is at most $\Lambda(\Theta_j, \Theta')$.

Similarly,

$$\begin{aligned}
b_j^i &\leq \sum_{\Theta' \in \mathcal{T}^{>}} Pr\left[\frac{1}{i}\Delta(\Theta_j, \Theta', i) < \epsilon - \epsilon_i(\Theta')\right] \\
&\leq \sum_{\Theta' \in \mathcal{T}^{>}} Pr\left[\frac{1}{i}\Delta(\Theta_j, \Theta', i) \leq (\mathrm{C}(\Theta') - \mathrm{C}(\Theta_j)) - \epsilon_i(\Theta')\right] \\
&\leq \sum_{\Theta' \in \mathcal{T}^{>}} e^{-2i\left(\frac{-\epsilon_i(\Theta')}{\Lambda(\Theta_j,\Theta')}\right)^2} \\
&\leq |\mathcal{T}^{>}| \frac{\delta_j}{2(L_j-1)|\mathcal{T}[\Theta_j]|}
\end{aligned}$$

In a similar manner, we can bound

$$c_j \leq \sum_{\Theta' \in \mathcal{T}^{<}} Pr\left[\frac{1}{L_j}\Delta(\Theta_j, \Theta', L_j) \geq \frac{\epsilon}{2}\right]$$

$$
\begin{aligned}
&\leq \sum_{\Theta' \in \mathcal{T}^<} e^{-2L_j \left(\frac{\epsilon/2}{\Lambda(\Theta_j, \Theta')}\right)^2} \\
&\leq |\mathcal{T}^<| \, exp\left\{ -2 \left(2 \left(\frac{\Lambda[\Theta_j]}{\epsilon} \right)^2 \ln \frac{2\,|\mathcal{T}[\Theta_j]|}{\delta_j} \right) \left(\frac{\epsilon}{2\,\Lambda[\Theta_j]} \right)^2 \right\} \qquad (14) \\
&= |\mathcal{T}^<| \frac{\delta_j}{2\,|\mathcal{T}[\Theta_j]|}
\end{aligned}
$$

(Line 14 uses the fact that $\Lambda(\Theta_j, \Theta') \leq \Lambda[\Theta_j]$); and likewise

$$
d_j \quad \leq \quad \sum_{\Theta' \in \mathcal{T}^>} Pr\left[\frac{1}{L_j} \Delta(\,\Theta_j,\, \Theta',\, L_j\,) \,<\, \frac{\epsilon}{2} \right] \quad \leq \quad \sum_{\Theta' \in \mathcal{T}^>} e^{-2L_j \left(\frac{\epsilon/2}{\Lambda(\Theta_j, \Theta')}\right)^2} \quad \leq \quad |\mathcal{T}^>| \frac{\delta_j}{2\,|\mathcal{T}[\Theta_j]|}
$$

Hence, the probability of making any type of mistake at the j^{th} stage is bounded by

$$
\left[\sum_{i=1}^{L_j - 1} a_j^i + b_j^i \right] + c_j + d_j \quad \leq \quad (L_j - 1) \frac{\delta_j}{2\,(L_j - 1)\,|\mathcal{T}[\Theta_j]|} \left(|\mathcal{T}^<| + |\mathcal{T}^>| \right) + \frac{\delta_j}{2\,|\mathcal{T}[\Theta_j]|} \left(|\mathcal{T}^<| + |\mathcal{T}^>| \right) \quad \leq \quad \delta_j
$$

as $\mathcal{T}^<$ and $\mathcal{T}^>$ are disjoint subsets of $\mathcal{T}[\Theta_j]$, and so $|\mathcal{T}^<| + |\mathcal{T}^>| \leq |\mathcal{T}[\Theta_j]|$.

The probability that PALO_1 will make a mistake (of any kind) on any step is at most

$$
\sum_{j=1}^{\infty} \left[\sum_{i=1}^{L_j - 1} a_j^i + b_j^i \right] + c_j + d_j \quad \leq \quad \sum_{j=1}^{\infty} \delta_j \quad = \quad \sum_{j=1}^{\infty} \frac{6\;\delta}{j^2 \pi^2} \quad = \quad \frac{6\;\delta}{\pi^2} \sum_{j=1}^{\infty} \frac{1}{j^2} \quad = \quad \frac{6\;\delta}{\pi^2} \frac{\pi^2}{6} \quad = \quad \delta
$$

as desired.

To deal with PALO_1's efficiency: Notice it can stay at any Θ_j performance element for at most L_j samples, a quantity that is clearly polynomial in $|\mathcal{T}[\Theta_j]| \leq |\mathcal{T}|$, $\Lambda[\Theta_j] \leq 2\lambda(c, \mathcal{Q}, \mathcal{S}_\Theta)$, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$.

To show that PALO_1 will terminate with probability 1 when $|\mathcal{S}_\Theta|$ is finite, notice that the only way that PALO_1 can fail to terminate here is if it cycles infinitely often: each time thinking first that some Θ_i is strictly better than Θ_j and so switching to it, and later, thinking that Θ_i is better, switching back. Of course, one of these inequalities is necessarily false; and the probability that PALO_1 will make such a mistake is bounded by $\max\{a_j^i, c_j\}_{i,j} \leq \frac{\delta_i}{2} < \frac{\delta}{2}$. The probability that PALO_1 will make this type of mistake an infinite number of times is therefore (at most) $\lim_{m \to \infty} \prod_{i=1}^{m} \frac{\delta}{2} = 0$. □ (Theorem 1)

Definition 1 (The K-WEAK Task)
INSTANCE: *A propositional theory* Σ *(in CNF), a positive integer* K*, a sample* $\mathcal{S} = \{q_i\}$*, and an integer* $f \in [0..K]$.

QUESTION: *Is there a horn weakening of* Σ *of size at most* K *which covers at least* f *of* $\mathcal{S}$*'s queries? (I.e., is there a conjunction of at most* K *horn clauses,* W*, where* $\Sigma \models W$*, such that* $|\{\, q \in S :\; W \models q \,\}| \geq f$*?)* □

Theorem 2 *The* K-WEAK *task is NP-complete.*

Proof: K-WEAK is clearly in NP, as we need only guess a potential weakening, and confirm its coverage. To show K-WEAK is NP-hard, we reduce the NP-complete SAT task to it [23]: Given any boolean formula for SAT, let Σ be the set of its clauses, and let $K = 2$, $f = 1$ and the set of sample $\mathcal{S} = \{\ q\&\neg q\ \}$ be a single unsatisfiable proposition. If Σ is consistent, then every weakening W is necessarily consistent, and hence has coverage 0; *i.e.*, $W \not\models q\&\neg q$. Otherwise, if Σ is inconsistent, then it will have a size-2 weakening of the form $\{r,\ \neg r\}$, whose categoricity is 1. Hence, any algorithm that can solve arbitrary instances of K-WEAK can also solve arbitrary SAT instances. $\Box$ (Theorem 2)

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *FOCS*, 1992.

[2] P. Auer, R. Holte, and W. Maass. Theory and applications of agnoistic PAC-learning with small decision trees. In *Proceedings of the Twelfth International Machine Learning Conference*, 1995.

[3] D. A. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, London, 1985.

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

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

[6] L. Booker, D. Goldberg, and J. Holland. Classifier systems and genetic algorithms. *Artificial Intelligence*, 40(1–3):235–282, Sept. 1989.

[7] A. Borgida and D. Etherington. Hierarchical knowledge bases and efficient disjunctive reasoning. In *Proceedings of KR-89*, pages 33–43, Toronto, May 1989.

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

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

[10] R. Caruana and D. Freitag. Greedy attribute selection. In *Proceedings of the Eleventh International Machine Learning Conference*, pages 28–36, N.J., 1994. Morgan Kaufmann.

[11] H. 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.

[12] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer-Verlag, New York, 1981.

[13] W. W. Cohen. Learning from textbook knowledge: A case study. In *Proceeding of AAAI-90*, 1990.

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

[15] M. Dalal and D. Etherington. Tractable approximate deduction using limited vocabulary. In *Proceedings of CSCSI-92*, Vancouver, May 1992.

[16] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54, Aug. 1988.

[17] G. DeJong. AAAI workshop on Explanation-Based Learning. Sponsored by AAAI, 1988.

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

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

[20] D. W. Etherington, A. Borgida, R. J. Brachman, and H. Kautz. Vivid knowledge and tractable reasoning: Preliminary report. In *Proceedings of IJCAI-89*, pages 1146–52, 1989.

[21] O. Etzioni. Tractable decision-analytic control. In *Proceedings of KR-89*, Toronto, May 1989.

[22] P. Fong. A quantitative study of hypotheis selection. In *Proceedings of the Twelfth International Machine Learning Conference*, July 1995.

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

[24] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[25] A. Goldberg. An average case complexity analysis of the satisfiability problem. In *Proceedings of the 4th Workshop on Automated Deduction*, pages 1–6, Austin, TX, 1979.

[26] I. J. Good. Twenty-seven principles of rationality. In V. P. Godambe and D. A. Sprott, editors, *Foundations of Statistical Inference.* Holt, Rinehart and Winston, Toronto, 1971.

[27] J. Gratch, S. Chien, and G. DeJong. Improving learning performance through rational resource allocation. In *Proceedings of AAAI-94*, pages 576–581, Seattle, WA, 1994.

[28] J. Gratch and G. DeJong. A hybrid approach to guaranteed effective control strategies. In *Proceedings of IWML-91*, pages 509–513, Evanston, Illinois, June 1991.

[29] J. Gratch and G. Dejong. COMPOSER: A probabilistic solution to the utility problem in speed-up learning. In *Proceedings of AAAI-92*, 1992.

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

[31] R. Greiner. PALO algorithms. Technical report, Siemens Corporate Research, 1993.

[32] R. Greiner. The challenge of revising impure theories. In *Proceedings of the Twelfth International Machine Learning Conference*, 1995.

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

[34] R. Greiner and C. Elkan. Measuring and improving the effectiveness of representations. In *Proceedings of IJCAI-91*, pages 518–24, Sydney, Australia, Aug. 1991.

[35] R. Greiner and R. Isukapalli. Learning to select useful landmarks. *IEEE Transactions on Systems, Man and Cybernetics*, accepted subject to modifications.

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

[37] R. Greiner and P. Orponen. Probably approximately optimal derivation strategies. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of KR-91*, Rochester, NJ, Apr. 1991. Morgan Kaufmann.

[38] R. Greiner and P. Orponen. Probably approximately optimal satisficing strategies. *Artificial Intelligence*, Fall 1995. to appear.

[39] R. Greiner and D. Schuurmans. Learning useful horn approximations. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of KR-92*, San Mateo, CA, Oct. 1992. Morgan Kaufmann.

[40] B. Grosof. Generalizing prioritization. In *Proceedings of KR-91*, pages 289–300, Boston, Apr. 1991.

[41] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In *Proceedings of AAAI-86*, pages 328–333, Philadelphia, Aug. 1986.

[42] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, pages 177–221, 1988.

[43] D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100:78–150, 1992.

[44] G. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1–3):185–234, Sept. 1989.

[45] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, Mar. 1963.

[46] E. J. Horovitz. Reasoning about beliefs and actions under computational resource constraints. In *Uncertainty in Artificial Intelligence 3*, Amsterdam, 1987. North Holland.

[47] T. Imielinski. Domain abstraction and limited reasoning. In *Proceedings of IJCAI-87*, pages 997–1003, Milan, Aug. 1987.

[48] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Machine Learning Conference*, pages 121–129, N.J., 1994. Morgan Kaufmann.

[49] I. Jurišica. Query optimization for knowledge base management systems; A machine learning approach. Master's thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, 1992.

[50] L. P. Kaelbling. *Learning in Embedded Systems.* MIT Press, 1993.

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

[52] R. M. Keller. Defining operationality for explanation-based learning. In *Proceedings of AAAI-87*, pages 482–87, Seattle, July 1987.

[53] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[54] H. Kyburg. The reference class. *Philosophy of Science*, 50, 1982.

[55] J. E. Laird, A. Newell, and P. S. Rosenbloom. SOAR: An architecture of general intelligence. *Artificial Intelligence*, 33(3), 1987.

[56] H. Levesque. Making believers out of computers. *Artificial Intelligence*, 30(1):81–108, Oct. 1986.

[57] R. Loui. Computing reference classes. In *AAAI Workshop on Uncertainty.* Morgan Kaufmann, St Paul, 1988.

[58] O. Maron and A. Moore. Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. In *Advances in Neural Information Processing Systems 6.* Morgan Kaufmann, April 1994.

[59] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach.* Kluwer Academic Publishers, Hingham, MA, 1988.

[60] S. Minton, J. Carbonell, C. Knoblock, D. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: A problem solving perspective. *Artificial Intelligence*, 40(1-3):63–119, Sept. 1989.

[61] T. M. Mitchell. The need for bias in learning generalizations. Technical Report CBM-TR-117, Laboratory for Computer Science Research, May 1980.

[62] T. M. Mitchell, S. Mahadevan, and L. I. Steinberg. LEAP: A learning apprentice for VLSI design. In *Proceedings of IJCAI-85*, pages 573–80, Los Angeles, Aug. 1985.

[63] P. Morris. Curing anomalous extensions. In *Proceedings of AAAI-87*, pages 437–42, Seattle, July 1987.

[64] K. S. Narendra and M. A. L. Thathachar. *Learning automata: an introduction.* Prentice Hall, Englewood Cliffs, N.J., 1989.

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

[66] D. Poole, R. Goebel, and R. Aleliunas. Theorist: A logical reasoning system for default and diagnosis. In N. Cercone and G. McCalla, editors, *The Knowledge Frontier: Essays in the Representation of Knowledge*, pages 331–52, New York, 1987. Springer Verlag.

[67] T. C. Przymusiński. On the declarative semantics of stratified deductive databases and logic programs. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 193–216, Los Altos, CA, 1987. Morgan Kaufmann Publishers, Inc.

[68] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

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

[70] S. Russell, D. Subramanian, and R. Parr. Provably bounded optimal agents. In *Proceedings of IJCAI-93*, Chamberry, Aug. 1993.

[71] S. J. Russell and B. N. Grosof. A declarative approach to bias in concept learning. In *Proceedings of AAAI-87*, pages 505–10, Seattle, WA, July 1987.

[72] B. Selman and H. Kautz. Knowledge compilation using horn approximations. In *Proceedings of AAAI-91*, pages 904–09, Anaheim, Aug. 1991.

[73] L. Shastri. Default reasoning in semantic networks: A formalization of recognition and inheritance. *Artificial Intelligence*, 39:283–355, 1989.

[74] H. A. Simon and J. B. Kadane. Optimal problem-solving search: All-or-none solutions. *Artificial Intelligence*, 6:235–247, 1975.

[75] D. E. Smith. Controlling backward inference. *Artificial Intelligence*, 39(2):145–208, June 1989.

[76] R. S. Sutton, editor. *Machine Learning*, volume 8. Kluwer Academic Publishers, 1992. Special Issue on Reinforcement Learning.

[77] P. E. Utgoff. *Shift of Bias for Inductive Concept Learning.* PhD thesis, Rutgers, Laboratory for Computer Science Research, Oct. 1984.

[78] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.

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

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

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

[82] J. Wong. Improving the accuracy of a representational system, May 1991. Bachelors Thesis, University of Toronto.