Learning to Reason

Learning to Reason*

Roni Khardon[†] **Dan Roth**[‡]

Aiken Computation Laboratory,
Harvard University,
Cambridge, MA 02138.
{roni,danr}@das.harvard.edu

January 1994
(Revised June 30, 1994)

Abstract

We introduce a new framework for the study of reasoning. The Learning (in order) to Reason approach developed here combines the interfaces to the world used by known learning models with the reasoning task and a performance criterion suitable for it. In this framework the intelligent agent is given access to her favorite learning interface, and is also given a grace period in which she can interact with this interface and construct her representation KB of the world W. Her reasoning performance is measured only after this period, when she is presented with queries α from some query language, relevant to the world, and has to answer whether W implies α.

The approach is meant to overcome the main computational difficulties in the traditional treatment of reasoning which stem from its separation from the "world". First, by allowing the reasoning task to interface the world (as in the known learning models), we avoid the rigid syntactic restriction on the intermediate knowledge representation. Second, we make explicit the dependence of the reasoning performance on the input from the environment. This is possible only because the agent interacts with the world when constructing her knowledge representation.

We show how previous results from learning theory and reasoning fit into this framework and illustrate the usefulness of the Learning to Reason approach by exhibiting new results that are not possible in the traditional setting. First, we give a Learning to Reason algorithm for a class of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. Second, we exhibit a Learning to Reason algorithm for a class of propositional languages that is not known to be learnable in the traditional sense.

---

*This paper is available from the Center for Research in Computing Technology, Division of Applied Sciences, Harvard University as technical report TR-02-94. A short version of the paper appears in the Proceedings of the National Conference on Artificial Intelligence, AAAI-94.

[†]Research supported by grant DAAL03-92-G-0164 (Center for Intelligent Control Systems).

[‡]Research supported by NSF grant CCR-92-00884 and by DARPA AFOSR-F4962-92-J-0466.

1 Introduction

Consider a baby robot, starting out its life. If it were a human being, nature would have provided for the infant a safe environment to spend an initial time period in. In this period she adapts to her environment and learns about the structures, rules, meta-rules, superstitions and other information the environment provides for. In the meantime, the environment protects her from fatal events. Only after this "grace period", she is expected to have "full functionality" (whatever that means) in her environment, but it is expected that her performance depends on the world she grew up in and reflects the amount of interaction she has had with it.

Realizing that learning is a central aspect of cognition, computational learning theory, a subfield concerned with modeling and understanding learning phenomena [Val84], takes a similar view in that the performance of the learner is measured only after the learning period, and with respect to the world. Traditional theories of intelligent systems, however, have assumed that cognition (namely, computational processes like reasoning, language recognition, object identification and other "higher level" cognitive tasks) can be studied separately from learning (See [Kir91] for a discussion of this issue.).

Reasoning in intelligent systems is studied under the knowledge-based system approach [McC58]. It is assumed that the knowledge is given to the system, stored in some *representation language* with a well defined meaning, and that there is some reasoning mechanism, used to determine what can be inferred from the sentences in the KB. The question of how this knowledge might be acquired and whether this should influence how the performance of the reasoning system is measured is not considered. The intuition behind this approach is based on the following constructive theorem:

Theorem 1.1 *If there is a learning procedure that can learn an exact description of the world in representation R, and there is an algorithm that can reason exactly using R, then there is a system that can learn to reason using R.*

We believe that the separate study of learning and the rest of cognition is, at least partly, motivated by the false assumption that the converse of Theorem 1.1 also holds, namely, that if there is a system that can learn to reason, then there is a learning procedure that can learn a representation of the world, and a reasoning procedure that can reason with it.

Computational considerations, however, render this self-contained reasoning approach as well as other variants of it not adequate for common-sense reasoning. This is true not only for the task of deduction, but also for many other forms of reasoning which have been developed, partly in order to avoid the computational difficulties in exact deduction and partly to meet some (psychological and other) plausibility requirements. All those were shown to be even harder to compute than the original formulation [Sel90, Pap91, Rot93]. The same is true for a lot of recent work in reasoning that aims at identifying classes of limited expressiveness, with which one can perform some sort of reasoning efficiently [BL84, Lev92, Sel90]. None of these works meet the strong tractability requirements required for common-sense reasoning (See, e.g., [Sha93]), even though the inference deals with limited expressiveness and is sometimes restricted in implausible ways (See, e.g., [DP91]).

An additional motivation for this work is the current disconnected state of the results in the fields of learning and reasoning. Perhaps the most important open question in learning theory today is that of learning DNF or CNF formulas (the problems are equivalent in the current framework). However, even if one had a positive result for learnability of these classes, this would be relevant only for classification tasks, and cannot be used for reasoning. The reason is that if the output of the learning algorithm is a CNF expression, then it cannot be used for reasoning, since this problem is computationally hard. From a traditional reasoning point of view, on the other hand, learning

a DNF is not considered interesting, since it does not relate easily to a rule based representation. In this work, therefore, while we build on the framework and some of the results of computational learning theory, we distinguish this, now traditional, learning task which we call here *Learning to Classify* from the new learning task, *Learning to Reason*.

Our Approach:

We argue that the main difficulties in the traditional treatment of reasoning stem from its separation from the "world". The effect of this is two folded:

(1) *Syntactic:* Since the reasoning task is a "stand alone" process, one has to define a rigid representation language with which reasoning problems are presented to the reasoner [Bro91].

(2) *Performance:* The performance criterion for the behavior of an intelligent system is irrespective of the world it functions in.

In this paper we develop a new framework for the study of Reasoning. Our approach differs from other approaches to reasoning in that it sees learning as an integral part of the process. The *Learning to Reason* theory developed here is concerned with studying the entire process of *learning* a knowledge base representation and *reasoning* with it.

Our model directly deals with the two problems mentioned above. The first is handled by bypassing the intermediate knowledge representation stage - we do not enforce the agent to keep her knowledge in a specific representation language[1]. The second problem is dealt with by making explicit the dependence of the reasoning performance on the input from the world. This is possible only because the agent interacts with the world when constructing the knowledge representation. Moreover, we take the view that a reasoner need not answer efficiently *all* possible queries, but only those that are "relevant", or "common", in a well defined sense. In particular, in this framework the intelligent agent is given access to her favorite learning interface, and is also given a grace period in which she can interact with this interface and construct her representation KB of the world W. Her reasoning performance is measured only after this period, when she is presented with queries α from some query language, relevant to the world, and has to answer whether W implies α.

We prove the usefulness of the Learning to Reason approach by showing that through interaction with the world, the agent truly gains additional reasoning power, over what is possible in the traditional setting. First, we give a Learning to Reason algorithm for a class of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. Second, we exhibit a Learning to Reason algorithm for a class of propositional languages, that is not known to be learnable in the traditional sense.

Furthermore, an inherent feature of the learning to reason approach, is a non-monotonic reasoning behavior it exhibits as a side effect, by using reasoning mistakes to improve the knowledge base. This desirable phenomena is hard to formalize, when dealing with reasoning systems defined independent of learning.

This work is similar in nature to the Neuroidal model developed by Valiant [Val94]. The model developed there provides a more comprehensive approach to cognition, and akin to our approach it views learning as an integral and crucial part of the process. There, the reasoner reasons from a learned knowledge base, a complex circuit, and thus can be modeled by our framework. Indeed reasoning in the Neuroidal model shares many properties with the Learning to Reason framework.

[1] We stress that the assumption that an intelligent agent has to keep her knowledge in some representation and use it when she reasons is basic to this framework. We allow the reasoner, however, to choose her own representation and even to use different representations for different tasks.

Our approach is different in that in an effort to give a more formal treatment of a reasoner that has learned her knowledge base, we currently restrict our discussion to a fixed, consistent world.

1.1 Summary of Results

To motivate our approach we first develop a sampling approach to reasoning that exemplifies the power gained by giving the agent access to the "world" she is supposed to reason in later. We prove that Learning to Reason is possible for arbitrary worlds and query languages under some technical restriction on the queries asked. However, for various considerations which we discuss later in the paper this approach alone is not sufficient as a model for reasoning, so we go on to define the Learning to Reason framework to capture those.

We start by studying the relation of the Learning to Reason (L2R) framework to the two existing ones, the traditional reasoning and the traditional learning (learning to classify (L2C)), and show that when the class of queries is unrestricted, L2R implies L2C. This is not the case, though, if the class of queries is restricted in some natural way.

We then discuss how existing results from learning and reasoning can be used in the new framework. Theorem 1.1 suggests a way to build a L2R system if one has an exact L2C algorithm and can reason exactly from its output representation. We discuss the case where one (or both) of these algorithms does not perform the exact task but rather some approximation of it. In particular we discuss the combination of PAC learning algorithms and mistake bound learning algorithms with (exact and approximate) reasoning algorithms.

We then consider Learning to Reason algorithms that use models (satisfying assignments) as the knowledge representation language. A characterization of reasoning with models for Horn theories was given in [KKS93] and for general theories in [KR94c]. We build on these results to exemplify the usefulness of the new approach:

- Consider the reasoning problem $W \models \alpha$, where W is some CNF formula and α is a $\log n$CNF (i.e., a CNF formula with at most $\log n$ literals in each clause). Then, when W has a polynomial size DNF[2] there is an exact and efficient Learning to Reason algorithm for this problem, while the traditional reasoning problem (with a CNF representation as the input) is NP-Hard.

- Consider the reasoning problem $W \models \alpha$, where W is any Boolean formula with a polynomial size DNF and α is a $\log n$CNF. Then, there is an exact and efficient Learning to Reason algorithm for this problem, while the class of Boolean formulas with polynomial size DNF is not known to be learnable in the traditional (Learning to Classify) sense.

These results show that neither a traditional reasoning algorithm (from the CNF representation) nor a traditional learning algorithm (that can "classify" the world) is necessary for Learning to Reason. Moreover, the results exemplify and aid in formalizing the notion of "intelligence is in the eye of the beholder" [Bro91], since our agent seems to behave logically, even though her knowledge representation need not be a logical formula and she does not use any logic or "theorem proving".

We exhibit another instance of Learning to Reason in which (formula-based) reasoning is computationally hard. We show that a known algorithm for learning Horn theories [FP93] can be used to learn the least upper bound of any theory, in a Horn representation, and therefore Learns to Reason with Horn queries.

The rest of this paper is organized as follows. In Section 2 we give the necessary background from reasoning and learning theory. In Section 3 we define the new Learning to Reason framework.

[2]The DNF representation is not given to the reasoner. Its existence is essential, since the algorithm is polynomial in its size.

In Section 4 we discuss the relation between learning to reason and learning to classify. In Section 5 we discuss how the traditional learning results and reasoning results can be used to get Learning to Reason algorithms. In Section 6 we discuss how model-based reasoning can be used for Learning to Reason. In Section 7 we consider another application of the Learning to Reason approach. We conclude with some reference to future work.

2 Preliminaries

We consider a Boolean function $f : \{0,1\}^n \to \{0,1\}$. The elements in the set $\{x_1, \ldots, x_n\}$ are called variables. Assignments in $\{0,1\}^n$ are denoted by x, y, z, and $weight(x)$ denotes the number of 1 bits in the assignment x. A literal is either a variable x_i (called a positive literal) or its negation $\overline{x_i}$ (a negative literal). A clause is a disjunction of literals and a CNF formula is a conjunction of clauses. For example $(x_1 \vee \overline{x_2}) \wedge (x_3 \vee \overline{x_1} \vee x_4)$ is a CNF formula with two clauses. A term is a conjunction of literals and a DNF formula is a disjunction of terms. For example $(x_1 \wedge \overline{x_2}) \vee (x_3 \wedge \overline{x_1} \wedge x_4)$ is a DNF formula with two terms. A CNF formula is Horn if every clause in it has at most one positive literal. We note that every Boolean function has many possible representations and in particular, both a CNF representation and a DNF representation. The size of CNF and DNF representation are, respectively, the number of clauses and the number of terms in the representation.

An assignment $x \in \{0,1\}^n$ satisfies f if $f(x) = 1$. Such an assignment x is also called a model of f. By "f implies g", denoted $f \models g$, we mean that every model of f is also a model of g. Throughout the paper, when no confusion can arise, we identify a Boolean function f with the set of its models, namely $f^{-1}(1)$. Observe that the connective "implies" ($\models$) used between Boolean functions is equivalent to the connective "subset or equal" ($\subseteq$) used for subsets of $\{0,1\}^n$. That is, $f \models g$ if and only if $f \subseteq g$.

2.1 Reasoning

A widely accepted framework for reasoning in intelligent systems is the knowledge-based system approach [McC58]. Knowledge, in some *representation language* is stored in a *Knowledge Base* (KB) that is combined with a reasoning mechanism. Reasoning is abstracted as a deduction task[3] of determining whether a *query* α, assumed to capture the situation at hand, is implied from KB (denoted KB $\models \alpha$). The discussion in this paper is restricted to propositional knowledge bases[4]. Let $\mathcal{F}, \mathcal{Q}$ be two arbitrary classes of representations for Boolean functions.

Definition 2.1 *An algorithm A is an* exact reasoning *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if for all $f \in \mathcal{F}$ and for all $\alpha \in \mathcal{Q}$, when A is presented with input (f, α), A runs in time polynomial in n and the size of f and α, and answers "yes" if and only if $f \models \alpha$.*

In the definition above the complexity of the algorithm depends on the size of the input, the formulas f and α. For this reason we usually restrict the discussion to a polynomial size representation of the functions considered. In particular, the class $\mathcal{F} =$ CNF denotes those Boolean functions with a polynomial size CNF, and $\mathcal{F} =$ CNF $\cap$ DNF denotes those Boolean functions with a polynomial size CNF, and a polynomial size DNF.

[3] We restrict ourselves here to deduction although the approach developed is applicable for other reasoning tasks, e.g., Bayesian networks.

[4] A propositional expression is just a Boolean function, and a propositional language is a class of Boolean functions. These terms are used in the reasoning and learning literature accordingly, and we use them interchangeably.

Answering the question KB $\models \alpha$ is equivalent to solving satisfiability for the formula KB $\wedge\overline{\alpha}$. This implies that if KB is given as a CNF or α is given as a DNF the problem is NP-Hard. It is also known that if KB is given as a DNF and α is given as a CNF the problem can be solved in polynomial time, but this representation was less favored than the previous ones possibly because of the belief that the KB should represent a set of rules (that easily translate to a CNF representation but not to a DNF), or since it is more difficult to update the representation in this form. Thus, when KB is given as a CNF, exact reasoning can be done efficiently only when satisfiability can be solved efficiently (e.g., Horn theories, CNF with clauses of length two).

2.2 Learning to Classify

The formal study of learning, (studied in computational learning theory [Val84, Hau87, Ang92]), abstracts the problem of inductively learning a concept as the problem of learning a Boolean function, given some access to an oracle that is familiar to some degree with the function. The interpretation is that the function's value is 1 when the input belongs to the target concept and 0 otherwise. The oracles are used to model the type of interface the learner may have to the world and they vary between learning models according to the amount of information we assume the learner receives about the concept. The learner is given some *grace period* of interaction with the world. Her performance is measured only after this grace period, when she is presented instances of the target concept and is supposed to classify them correctly (always or "most of the time", according to the model). Next we describe some oracles that are standard in learning theory, introduce others that are especially suited for Reasoning and define the learning problem.

Definition 2.2 *A* Membership Query Oracle *for a function f, denoted $MQ(f)$, is an oracle that when given an input $x \in \{0,1\}^n$ returns $f(x)$.*

Definition 2.3 *An* Equivalence Query Oracle *for a function f, denoted $EQ(f)$, is an oracle that when given as input a function g, answer "yes" if and only if $f \equiv g$. If it answers "no" it supplies a counterexample, namely, an $x \in \{0,1\}^n$ such that $f(x) \neq g(x)$. A counterexample x satisfying $f(x) = 1$ ($f(x) = 0$) is called a positive (negative) counterexample.*

Definition 2.4 *An* Example Oracle *for a function f, with respect to the probability distribution D, denoted $EX_D(f)$, is an oracle that when accessed, returns $(x, f(x))$, where x is drawn at random according to D.*

We note that these are not all the oracles a learner can use when learning a function. For example, the oracles "incomplete membership queries" [AS91] "faulty example oracles" [AL88] "malicious example oracles" [Val85, KL88], "entailment oracle" [FP93] are also studied in the literature and can be used here as well. The Reasoning oracles we introduce next model the case where an agent learns from mistakes it makes while reasoning.

Definition 2.5 *A* Reasoning Query Oracle *for a function f and a query language $\mathcal{Q}$, denoted $RQ(f, \mathcal{Q})$, is an oracle that when accessed performs the following protocol with a learning agent A. (1) The oracle picks an arbitrary query $\alpha \in \mathcal{Q}$ and returns it to A. (2) The agent A answers "yes" or "no" according to her belief with regard to the truth of the statement $f \models \alpha$. (3) If A's answer is correct then the oracle says "correct". If the answer is wrong the oracle answers "wrong" and in case $f \not\models \alpha$ it also supplies a counterexample (i.e., $x \in f \setminus \alpha$).*

Definition 2.6 *A* Strong Reasoning Query Oracle *for a function f and a query language $\mathcal{Q}$, denoted $SRQ(f, \mathcal{Q})$, is an oracle that when accessed performs the following protocol with a learning agent A. (1) The oracle picks an arbitrary query $\alpha \in \mathcal{Q}$ and returns it to A. (2) The agent A answers "yes" or "no" according to her belief with regard to the truth of the statement $f \models \alpha$. When it answers "no", A also returns an assignment y for which she claims that $y \in f \setminus \alpha$. (3) If A's answer is correct then the oracle says "correct". If the answer is wrong the oracle answers "wrong" and if $f \not\models \alpha$ it also supplies a counterexample (i.e. $x \in f \setminus \alpha$). In the case that $f \models \alpha$ the assignment y serves as a counterexample for the belief of A.*

We denote by $I(f)$ the *interface* available to the learner when learning f. This can be any subset of the oracles defined above, and might depend on some fixed but arbitrary and unknown distribution D over the instance space $\{0,1\}^n$.

We now define various performance criteria for learning algorithms. We distinguish between a "batch" type learning scenario, and an "on-line" scenario. In the "batch" scenario the learning algorithm interfaces with the environment, via $I(f)$, in order to acquire the skill of labeling future instances. The performance of the algorithm is measured only after some "grace period", that must be of length polynomial in the size of the concept learned (and also in the quality of its performance), when it is required to predict the value of f on some other example drawn according to D. The dependency of the algorithm on the size of the concept is not made explicit in the following definitions, assuming that it has some polynomial (in n) representation. We denote by $h(x)$ the prediction[5] of the algorithm on the example $x \in \{0,1\}^n$.

Definition 2.7 *An algorithm A is an* Exact Learn to Classify (E-L2C) *algorithm for a class of functions $\mathcal{F}$, if there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, when given access to $I(f)$, A runs in time $p(n)$ and then, given any $x \in \{0,1\}^n$, takes time $p(n)$ to predict σ such that $\sigma = f(x)$.*

Definition 2.8 *An algorithm A is a* Probably Approximately Correct Learn to Classify (PAC-L2C) *algorithm for a class of functions $\mathcal{F}$, if there exists a polynomial $p(,,)$ such that for all $f \in \mathcal{F}$, on input ϵ, δ and given access to $I(f)$, A runs in time $p(n, 1/\epsilon, 1/\delta)$ and then given any $x \in \{0,1\}^n$, predicts $h(x)$ in time $p(n, 1/\epsilon, 1/\delta)$. A's predictions have the property that with probability at least $1 - \delta$, $Prob_{x \in D}[f(x) \neq h(x)] < \epsilon$. The parameter δ is called the* confidence *of the algorithm A and ϵ its* accuracy.

In the on-line (or, mistake-bound) scenario, algorithm A is presented with a sequence of examples in $\{0,1\}^n$. At each stage, the algorithm is asked to predict $f(x)$ and is then told whether its prediction was correct. Each time the learning algorithm makes an incorrect prediction, we charge it one *mistake*.

Definition 2.9 *An algorithm A is a* Mistake Bound Learn to Classify (MB-L2C) *algorithm for a class of functions $\mathcal{F}$, if there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, for every (arbitrary infinite) sequence of instances, A runs in time $p(n)$ (on each example) and makes no more than $p(n)$ mistakes.*

It is known [Ang88, Lit89] that a Mistake bound learning algorithm can be transformed into a PAC learning algorithm, and that an E-L2C (or PAC-L2C) algorithm that uses the $EQ(f)$ oracle can be transformed into an algorithm that achieves PAC-L2C using an Example Oracle and without using $EQ(f)$.

[5] Sometimes an equivalent definition is used, in which the learning algorithm outputs an hypothesis h and its performance is measured with respect to it. In order not to make any assumptions on how this hypothesis is represented we assume here that it is internal to the algorithm.

3 Learning to Reason

In this section we motivate and introduce the definitions of the *Learning to Reason* model.

Let $W \in \mathcal{F}$ be a Boolean function that describes the world exactly. Let α be some Boolean function (a query) and let D be some fixed but arbitrary and unknown probability distribution over the instance space $\{0,1\}^n$. As in the learning framework, we assume that D governs the occurrences of instances in the world.

Definition 3.1 *The query α is called* legal *if $\alpha \in \mathcal{Q}$.*

Definition 3.2 *The query α is called* (W,ϵ)-fair *if either $W \subseteq \alpha$ or $Prob_D[W \setminus \alpha] > \epsilon$.*

The intuition behind this definition is that the algorithm is allowed to err in case $W \not\models \alpha$, but the weight of W outside α is very small. Along with ϵ, the *accuracy* parameter, we use a *confidence* parameter, δ, and sometimes might allow the reasoning algorithm to err, with small probability, less than δ.

3.1 A Sampling Approach

To motivate our approach we first consider the following simple approach to reasoning: whenever presented with a query α, first use the Example Oracle $EX_D(W)$ and take a sample of size $m = (1/\epsilon)\ln(1/\delta)$, where δ and ϵ are the required confidence and accuracy parameters. Then, perform the following model-based test: for all the samples $(x,1)$ sampled from $EX_D(W)$ (note that we ignore the samples labeled 0), check whether $\alpha(x) = 1$. If for some x, $\alpha(x) = 0$ say $W \not\models \alpha$; otherwise say $W \models \alpha$.

The following analysis shows that if α is (W,ϵ)-fair then with probability at least $1-\delta$ the algorithm is correct.

Clearly, if $W \models \alpha$ the algorithm never errs. The algorithm makes a mistake only if $W \not\models \alpha$ but an instance x in $W \cap \overline{\alpha}$ is never sampled. An instance x, picked randomly according by $EX_D(W)$ is in $W \cap \overline{\alpha}$ with probability greater than ϵ. Therefore, the probability that an instance $x \in W \cap \overline{\alpha}$ is missed in $m = \frac{1}{\epsilon}\ln(1/\delta)$ trials is less than

$$(1-\epsilon)^m = (1-\epsilon)^{\frac{1}{\epsilon}\ln(1/\delta)} < \delta. \tag{1}$$

Therefore, the algorithm errs on a (W,ϵ)-fair query with probability less than δ.

This analysis depends on the fact that the samples are independent of the query α, and therefore a different sample has to be taken for every query α. We call this a *repeated sampling* approach[6]. However, repeated sampling is not a plausible approach to reasoning in intelligent systems. When presented with a query, an agent cannot allow herself further interactions with the world before answering the query. Especially if the query is "A lion is approaching $\Rightarrow$ I have to run away".

For a more plausible approach, we now modify the above analysis and show that a *one time sampling* approach can also guarantee reasoning with respect to (W,ϵ)-fair queries, with confidence $1-\delta$.

From Eq.1 we have that the probability that *any* (W,ϵ)-fair query $\alpha \in \mathcal{Q}$, that is not implied by W is answered by "yes" after m samples is less than

$$|\mathcal{Q}|(1-\epsilon)^m, \tag{2}$$

[6] A similar, more sophisticated approach was developed in [Kea92] for the case in which both the knowledge base and the queries are learned concepts in the PAC sense. It is implicit there that for each possible query one needs a new sample.

where $|\mathcal{Q}|$ is the size of the class $\mathcal{Q}$ of queries. Therefore, if we take more than

$$m = \frac{1}{\epsilon}(\ln |\mathcal{Q}| + \ln \frac{1}{\delta}) \tag{3}$$

samples from $EX_D(f)$, by substituting m from Eq. 3 in Eq. 2 we have that the model-based test described above errs on every (W, ϵ)-fair query with probability less than δ. Since all the queries in $\mathcal{Q}$ are propositional formulas of polynomial size, the number m of samples required to guarantee this performance is polynomial. This approach is therefore feasible.

It is important not to confuse this approach of sampling from D, the distribution that governs the occurrences of instances in the world, with sampling approaches that aim at performing approximate reasoning from a fixed knowledge base (e.g., a formula-based knowledge base W). There, no access to the "world" $EX_D(W)$ is available, and the sampling is done in general from the uniform distribution, in order to find satisfying assignments of W. This approach, that is typical in many approximate reasoning techniques, is intractable in a very strong sense [Rot93, KR94b].

The new approach exemplifies the power gained by giving the reasoner access to the "world" she is supposed to reason in later. Learning to reason is possible for arbitrary world and query languages given that the queries are fair.

However, the one-time sampling approach is not the ultimate solution for reasoning and there are many reasons not to be satisfied with this approach as the sole solution for the reasoning problem. We briefly discuss some of those reasons as a motivation for the approach taken in the rest of the paper.

Exact reasoning: In many cases it is natural to require the reasoning to be *exact* with respect to the world. Unlike the sampling approach other representations might support it.

Active learning: Certain events we want to reason about might be too rare to have a significant weight under D, the distribution that governs the occurrences of instances in the world (i.e., they might not be "fair"). To facilitate this, we might want to supply the reasoner with a richer class of interfaces with the world. Oracles that seem to be relevant in this context are partial assignments oracles [Val84] more selective oracles [Ams88], and the use of declarative information. The availability and plausibility of various oracles as well as the technical investigation into algorithms and knowledge representations that can make use of these oracles are the main questions here. Preliminary progress in this direction is done in [KR94a].

Incremental/Non-monotonic nature of Reasoning: Reasoning with an inductively learned knowledge base yields a desirable non-monotonic behavior: every time the algorithm makes a reasoning mistake, it changes its mind (i.e., the underlying learning algorithm does), learns something about the world, and would not make the same type of mistake again. More importantly, unlike reasoning from a large set of randomly selected examples, the algorithms we present later in this paper for reasoning from an inductively learned knowledge base, have the property that while exhibiting a non-monotonic behavior, they also converge (after polynomially many reasoning mistakes) to yield an exact reasoning behavior. We would like to investigate which algorithms and knowledge base representations yield this behavior.

Space consideration: The one-time sampling approach suggested above might require maintaining very large (yet polynomial) knowledge base, and accordingly, a long period of interfacing with the world, before it can guarantee acceptable performance. There might be other, much smaller representations that perform as well.

3.2 Learning to Reason: Definitions

We now provide the basic definitions that allow us to initiate the formal study into the questions discussed above. As pointed out above, our definitions do not allow a *repeated sampling* approach, but do allow *one-time sampling*.

As in the case of Learning to Classify we distinguish between Learning to Reason in a "batch" type scenario, and an "on-line" Learning to Reason.

In the batch scenario, the algorithm interfaces with the environment, via $I(f)$, in order to acquire the *reasoning skill* for answering future queries. The performance of the algorithm is measured only after some "grace period", that must be of length polynomial in the size of the world description (and in the quality of its performance), when it is required to decide, without interfacing the world again, if some query is entailed by f. As before we define an exact version and a non-exact version.

Definition 3.3 *An algorithm A is an* Exact Learn to Reason (E-L2R) *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, given access to $I(f)$, A runs in time $p(n)$ and then, when presented with any query $\alpha \in \mathcal{Q}$, A runs in time $p(n)$, does not access $I(f)$, and answers "yes" if and only if $f \models \alpha$.*

Definition 3.4 *An algorithm A is a* Probably Approximately Correct Learn to Reason (PAC-L2R) *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if there exists a polynomial $p(,,)$ such that for all $f \in \mathcal{F}$, on input ϵ, δ, given access to $I(f)$, A runs in time $p(n, 1/\epsilon, 1/\delta)$ and then with probability at least $1 - \delta$, when presented with any (f, ϵ)-fair query $\alpha \in \mathcal{Q}$, A runs in time $p(n, 1/\epsilon, 1/\delta)$, does not access $I(f)$, and answers "yes" if and only if $f \models \alpha$.*

In the batch scenario above we did not allow access to $I(f)$ while in the query answering phase. In the on-line version however, we consider a query α given to the algorithm as if given by the reasoning oracle $RQ(f, \mathcal{Q})$ defined above. Thus, a reasoning error may supply the algorithm a counterexample which in turn can be used to improve its future reasoning behavior. We allow the L2R algorithm to access $I(f)$ during this update, but *not* while answering a query. In this on-line (or, mistake-bound) scenario, the L2R algorithm is charged one mistake each time the reasoning query is answered incorrectly.

Definition 3.5 *An algorithm A is a* Mistake Bound Learn to Reason (MB-L2R) *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if A interacts with the reasoning oracle $RQ(f, \mathcal{Q})$, and there exists a polynomial $p()$ such that for all $f \in \mathcal{F}$, (1) A runs in time $p(n)$ (on each query) and answers "yes" or "no" according to its belief with regard to the truth of the statement $f \models \alpha$, without accessing $I(f)$, (2) then runs in time $p(n)$ before it is ready for the next query (possibly, with accessing $I(f)$), and (3) for every (arbitrary infinite) sequence of queries, A makes no more than $p(n)$ mistakes.*

4 The relations between L2R and L2C

In this section we investigate the relation between Learning to Reason and Learning to Classify. Clearly, given an Exact-L2C algorithm A, the ability to reason (in the traditional sense) efficiently with the hypothesis A keeps is sufficient to yield an efficient Exact L2R algorithm. In this section we consider the other direction of this relation. That is, given an algorithm that can Learn to Reason, is it necessarily the case that there is an algorithm that can Learn to Classify ?

Intuitively, the classification task seems to be easier than the reasoning task. In the first we need to evaluate correctly a function on a single point, while in the latter we need to know if *all the models* of the function are also models of the query. It is not surprising therefore, that if *any*

subset of $\{0,1\}^n$ is a legal query, the ability to L2R implies the ability to L2C. We formalize this in the following theorem.

Let DISJ be the class of all disjunctions over n variables.

Theorem 4.1 *If there is an Exact-L2R algorithm for the reasoning problem $(\mathcal{F}, DISJ)$ then there is an Exact-L2C algorithm for the class $\mathcal{F}$.*

Proof: Observe that for a Boolean function f and a assignment $z \in \{0,1\}^n$,

$$f(z) = 0 \Leftrightarrow f \models \{0,1\}^n \setminus \{z\}. \tag{4}$$

Denote by d_z the disjunction define by $d_z = \bigvee x_i^{z_i}$ (where $x_i^0 = x_i, x_i^1 = \overline{x_i}$). For example, if $z = (1,0,0)$ then $d_z = \overline{x_1} \vee x_2 \vee x_3$. Clearly d_z satisfies $d_z(y) = 0$ if and only if $y = z$, and Eq. 4 can be written therefore as

$$f(z) = 0 \Leftrightarrow f \models d_z. \tag{5}$$

Now, given an Exact-L2R algorithm A for the reasoning problem $(\mathcal{F}, \text{DISJ})$ we use it to construct an Exact-L2C algorithm B for $\mathcal{F}$ as follows: B first runs A (enough time to guarantee A's performance), and when given a assignment $z \in \{0,1\}^n$ for classification by f, it first computes the disjunction d_z and gives it to A. B returns "1" if and only if A returns "no" on d_z. The correctness of the algorithm is clear from Eq. 5. ■

We note that reasoning with DISJ is as hard as reasoning with general CNF queries, since in general $f \models (\alpha \wedge \beta)$ if and only if $f \models \alpha$ and $f \models \beta$.

The proof of the above theorem does not go through if the class of queries $\mathcal{Q}$ does not include all of DISJ, and does not hold also in the non-exact L2R formulation. Strictly speaking, we leave open the question of whether there is a case, in which there exists a Learning to Reason algorithm for the problem $(\mathcal{F}, \mathcal{Q})$ but there is no Learning to Classify algorithm for $\mathcal{F}$. The reason is that in general, it is hard to prove that there *does not exist* a L2C algorithm. We can show however, that there is a class of Boolean functions $\mathcal{F}$, and a class of queries $\mathcal{Q}$, such that there exists a Learn to Reason algorithm for the problem $(\mathcal{F}, \mathcal{Q})$ but *it is not known* how to Learn to Classify $\mathcal{F}$. We defer the presentation of this result to a later section (Section 6.5).

5 L2R by Combining Learning and Reasoning

In this section we identify techniques for Learning to Reason. We show how the interaction with the world (i.e., learning) can be used to collect some approximate knowledge, and then how to use this knowledge to reason successfully. The results in this section combine results from computational learning theory with results from the theory of reasoning, and thus can be considered as a positive outcome of studying these problems separately. However, as we show below, the significance of the results presented in this section is that they exhibit the *limitations* of L2R by combining separate reasoning and learning algorithms.

The case of exact learning algorithms that, in addition, produce as output a representation that allows for efficient reasoning has already been discussed above. Here we consider the relaxation of these requirements.

5.1 Learning to Reason via PAC Learning

Assume that the world description W is in $\mathcal{F}$ and there is a PAC-L2C algorithm A for $\mathcal{F}$. We first show how a PAC learning algorithm, if it has an additional property, can be combined with a reasoning algorithm to yield a PAC-Learn to Reason algorithm.

Definition 5.1 *An algorithm that PAC learns to classify $\mathcal{F}$ is said to* learn $f \in \mathcal{F}$ from below *if, when learning f, the algorithm never makes mistakes on instances outside of f. (I.e., if h is the hypothesis the algorithm keeps then it satisfies $h \subseteq f$.)*

Theorem 5.1 *Let A be a PAC-Learn to Classify algorithm for the function class $\mathcal{F}$ and assume that A uses the class of representations $\mathcal{H}$ as its hypotheses. Then, if A learns $\mathcal{F}$ from below, and there is an exact reasoning algorithm B for the reasoning problem $(\mathcal{H}, \mathcal{Q})$, then there is a PAC-Learn to Reason algorithm C for the reasoning problem $(\mathcal{F}, \mathcal{Q})$.*

Proof: The learning algorithm C simply runs A and then uses B in order to reason with the hypothesis $h \in \mathcal{H}$ of A. Let α be a (W, ϵ)-fair legal query. Assume first that $W \models \alpha$. Then, since the algorithm A learns W from below its hypothesis h satisfies $h \subseteq W \subseteq \alpha$ and therefore $h \models \alpha$ and C answers correctly. When $W \not\models \alpha$, since α is (W, ϵ)-fair, we know that $Prob_D[W \setminus \alpha] > \epsilon$. Together with the fact that $Prob_D[W \setminus h] < \epsilon$, we have $h \setminus \alpha \neq \emptyset$ and therefore $h \not\models \alpha$, so again, C answers correctly. ■

The performance guaranteed by the above theorem is not better than the one-time sampling approach, while using a possibly more complicated algorithm. The result is significant, however, for the following reasons; (1) It shows the limitations of L2R by combining reasoning and learning algorithms: The theorem cannot be extended to the case where the the learning algorithm has also error from above (i.e., where it makes prediction mistakes also on instances of W). The reason is that if $h \setminus W \neq \emptyset$ then it might be the case that $W \models \alpha$ but $h \not\models \alpha$ exactly because of those additional assignments in h. (2) The theorem allows for the PAC learning algorithms to use any set of oracles (and in particular, other, more plausible interfaces to specific applications) while the sampling result holds only for the example oracle. (3) The theorem allows for a system that performs classification as well as reasoning from the same knowledge base, and finally, (4) The theorem explains the behavior of mistake bound algorithms discussed in the next section.

For a similar result to hold for an algorithm with two sided error one has to use a somewhat strong version of an "approximate deduction".

Definition 5.2 *An algorithm A is an* approximate deduction *algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if for every ϵ, and for all $f \in \mathcal{F}$ and $\alpha \in \mathcal{Q}$ when presented with input (f, α), A runs in time polynomial in n and $1/\epsilon$ and answers "yes" if and only if $Prob_D[f \setminus \alpha] < \epsilon$.*

Given this notion of deduction, we can now show:

Theorem 5.2 *Let A be a PAC-Learn to Classify algorithm for the function class $\mathcal{F}$ and assume that A uses the class of representations $\mathcal{H}$ as its hypotheses. If there is an approximate deduction algorithm B for the reasoning problem $(\mathcal{H}, \mathcal{Q})$, then there is a PAC-Learn to Reason algorithm C for the reasoning problem $(\mathcal{F}, \mathcal{Q})$.*

Proof: The algorithm C runs A, enough time to guarantee an ϵ-accurate hypothesis h. Then, given a query α, it uses the ϵ-approximate deduction algorithm B. We show that C succeeds on $(W, 2\epsilon)$-fair queries.

Suppose $W \models \alpha$. Then $Prob_D[W \setminus \alpha] = 0$, which implies $Prob_D[h \setminus \alpha] < \epsilon$ and the algorithm answers correctly, "yes".

In the other case, when $W \not\models \alpha$, since α is $(W, 2\epsilon)$-fair, we know that $Prob_D[W \setminus \alpha] > 2\epsilon$. Therefore $Prob_D[h \setminus \alpha] > \epsilon$ and the algorithm answers correctly, "no". ■

5.2 Learning to Reason via Mistake Bound Learning

Consider a Mistake Bound algorithm that keeps a hypothesis that allows for efficient reasoning. In this section we observe that in this case, the algorithm can be used to construct a Learn to Reason algorithm.

Let A be a Mistake Bound algorithm and assume it has been used long enough to guarantee PAC performance [Lit89]. In the case it has used up all of its mistakes on negative examples (i.e., on assignments outside of W), the hypothesis it uses is a "learn from below" hypothesis, and we can reason with it and succeed on all (W, ϵ)-fair queries.

Unfortunately, we cannot force the algorithm (or rather the interface) to make all these mistakes within the grace period. If we use an initial grace period to ensure its PAC properties then after the algorithm is ready to answer queries it may still make (a limited number of) mistakes. If the reasoning mistakes can be used as a source for negative counterexamples (i.e., assignments in $h \setminus W$) then after making a polynomial number of reasoning mistakes the algorithm would hold a hypothesis that approximates W from below, and thus supports exact reasoning.

This is the behavior we model with the reasoning oracle, $RQ(f, \mathcal{Q})$. Notice though, that while this is enough to guarantee exact reasoning, it is not enough to learn even a PAC approximation of W, since the algorithm receives only positive counterexamples. Using the strong oracle $SRQ(f, \mathcal{Q})$ instead, the algorithm is allowed to use also negative counterexamples and in this case the hypothesis will converge to an exact description of W.

It is interesting to note that reasoning with this type of an algorithm yields a non monotonic reasoning behavior. Every time the algorithm makes a reasoning mistake, it changes its mind, learns something about the world, and would not make the same mistake again. This is an inherent feature of the learning to reason approach, and it captures a phenomenon that is hard to formalize, when dealing with reasoning systems defined independent of learning.

6 Learning to Reason via Model Based Reasoning

In this section we develop the main technical results of this paper and exhibit the computational advantages of the Learning to Reason approach. We deviate from the traditional setting of "first learn to classify, then reason with the hypothesis": A learning algorithm is used first, but rather than learning a "classifying hypothesis", it constructs a knowledge representation that allows for efficient reasoning.

We show that this process yields two interesting outcomes, that were not possible in the traditional setting. First, we give a L2R algorithm for the class CNF $\cap$ DNF for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base. Second, we develop a L2R algorithm for the class of functions with polynomial size DNF, for which a L2C algorithm is not known. The results in this section use two recent results, one on learning via monotone theory [Bsh93] and the other on reasoning with models [KR94c]; we first introduce some definitions and results from there.

6.1 Monotone Theory

Definition 6.1 (Order) *We denote by* $\leq$ *the* usual partial order *on the lattice* $\{0,1\}^n$*, the one induced by the order* $0 < 1$*. That is, for* $x, y \in \{0,1\}^n$*,* $x \leq y$ *if and only if* $\forall i, x_i \leq y_i$*. For an assignment* $b \in \{0,1\}^n$ *we define* $x \leq_b y$ *if and only if* $x \oplus b \leq y \oplus b$ *(Here* $\oplus$ *is the bitwise addition modulo* 2*).*

Intuitively, if $b_i = 0$ then the order relation on the ith bit is the normal order; if $b_i = 1$, the order relation is reversed and we have that $1 <_{b_i} 0$. We now define:
The *monotone extension of* $z \in \{0,1\}^n$ with respect to b:

$$\mathcal{M}_b(z) = \{x \mid x \geq_b z\}.$$

The *monotone extension of* f with respect to b:

$$\mathcal{M}_b(f) = \{x \mid x \geq_b z, \ \textit{for some } z \in f\}.$$

The set of *minimal assignments of* f with respect to b:

$$\min_b(f) = \{z \mid z \in f, \ \textit{such that } \forall y \in f, z \not\geq_b y\}.$$

Every Boolean function f can be represented in the following form:

$$f = \bigwedge_{b \in B} \mathcal{M}_b(f) = \bigwedge_{b \in B} \bigvee_{z \in \min_b(f)} \mathcal{M}_b(z) \tag{6}$$

In the representation given in Equation 6, $B \subseteq \{0,1\}^n$ is called a *monotone basis* for f. It is known that the size of the basis is at most the CNF size of f.

The set of *floor* assignments of an assignment x, with respect to the order relation b, denoted $\lfloor x \rfloor_b$, is the set of all elements $z <_b x$ such that there does not exist y for which $z <_b y <_b x$ (i.e., z is strictly smaller than x relative to b and is different from x in exactly one bit).

The set of *local minimal assignments of* f with respect to b is:

$$\min_b^*(f) = \{x \mid x \in f, \text{ and } \forall y \in \lfloor x \rfloor_b, \ \ y \notin f\}.$$

Clearly we have that $\min_b(f) \subseteq \min_b^*(f)$. It is known that the size of $\min_b^*(f)$ and therefore the size of $\min_b(f)$, is bounded by the DNF size of f.

The representation in Equation 6 yields the following necessary and sufficient condition for $x \in \{0,1\}^n$ to be positive for f:

Corollary 6.1 *Let* B *be a basis for* f*,* $x \in \{0,1\}^n$*. Then,* $x \in f$ *(i.e.,* $f(x) = 1$*) if and only if for every basis element* $b \in B$ *there exists* $z \in min_b(f)$ *such that* $x \geq_b z$*.*

6.2 Restricted queries

A monotone basis can be used to characterize a class of Boolean functions: all those which have the same basis. Thus, we can use the notion of a basis to characterize classes of propositional queries:

Definition 6.2 *Let* B *be a monotone basis for the Boolean function* f*. A class* $\mathcal{Q}$ *of Boolean functions is* relevant to f *if* B *is also a monotone basis for all the functions in* $\mathcal{Q}$*.*

Definition 6.3 *A class $\mathcal{Q}$ of Boolean functions is* common *if $\mathcal{Q}$ has a fixed, polynomial size monotone basis.*

It is known ([KR94c]), that the class of Horn CNF functions has a basis of size $n+1$, and that the class of $\log n$ CNF functions (CNF in which the clauses contain at most $O(\log n)$ literals) has a basis of size less than n^3. Other important examples of common languages are: k-quasi-Horn queries (a generalization of Horn theories in which there are at most k positive literals in each clause), reverse-k-quasi-Horn queries and others. Clearly, the union of common classes is also common.

6.3 Reasoning with models

Reasoning with models is a very simple and highly parallelizable procedure. Let $\Gamma \subseteq f \subseteq \{0,1\}^n$ be a set of models. To decide whether $f \models \alpha$ use the model-based approach to deduction: for all the models $z \in \Gamma$ check whether $\alpha(z) = 1$. If for some z, $\alpha(z) = 0$ say "no"; otherwise say "yes". By definition, if $\Gamma = f$ this approach yields correct deduction, but representing f by explicitly holding *all* the possible models of f is not plausible. A model-based approach becomes feasible if Γ supports correct deduction and is small. In the following we characterize a model-based knowledge base that provides for correct reasoning.

Definition 6.4 *Let $\mathcal{F}$ be a class of functions, and let B be a basis for $\mathcal{F}$. For a knowledge base $f \in \mathcal{F}$ we define the set $\Gamma = \Gamma_f^B$ of* characteristic models *to be the set of all minimal assignments of f with respect to the basis B. Formally,*

$$\Gamma_f^B = \cup_{b \in B} \{ z \in min_b(f) \}.$$

The following is the basic theorem of the theory of reasoning with models:

Theorem 6.2 *Let f, $\alpha \in \mathcal{F}$ and let B be a basis for $\mathcal{F}$. Then $f \models \alpha$ if and only if for every $u \in \Gamma_f^B$, $\alpha(u) = 1$.*

Next we discuss the notion of a least upper bound of a Boolean function, its relation to the monotone theory and its usage in model-based reasoning.

Definition 6.5 (Least Upper-bound) *Let $\mathcal{F}, \mathcal{G}$ be families of propositional languages. Given $f \in \mathcal{F}$ we say that $f_{lub} \in \mathcal{G}$ is a $\mathcal{G}$-least upper bound of f if and only if $f \subseteq f_{lub}$ and there is no $f' \in \mathcal{G}$ such that $f \subset f' \subset f_{lub}$.*

Theorem 6.3 *Let f be any propositional theory and $\mathcal{G}$ a class of all propositional theories with basis B. Then*

$$f_{lub} = \bigwedge_{b \in B} \mathcal{M}_b(f).$$

The above theorem shows that the logical function represented by the set Γ_f^B, where B is a basis for $\mathcal{G}$, is the LUB of f in $\mathcal{G}$. Nevertheless, this representation is sufficient to support exact deduction with respect to queries in $\mathcal{G}$:

Theorem 6.4 *Let $f \in \mathcal{F}$, $\alpha \in \mathcal{G}$ and let B be a basis for $\mathcal{G}$. Then $f \models \alpha$ if and only if for every $u \in \Gamma_f^B$, $\alpha(u) = 1$.*

6.4 Learning to Reason without Reasoning

In this section we give a Learning to Reason algorithm for a class of propositional languages for which there are no efficient reasoning algorithms, when represented as a traditional (formula-based) knowledge base.

Theorem 6.5 *There is an Exact-Learn to Reason algorithm, that uses an Equivalence Query oracle and a Membership Query Oracle for the reasoning problem* $(CNF \cap DNF, \mathcal{Q})$*, where* $\mathcal{Q}$ *is any class of relevant and common queries.*

Proof: The Learning to Reason algorithm learns a model-based representation for the target function f and then uses model-based reasoning to answer queries with respect to it. In [Bsh93] an algorithm is developed that uses an Equivalence Query oracle and a Membership Query Oracle to learn an exact representation of any function $f \in CNF \cap DNF$. As a byproduct of this algorithm, the set of all minimal models with respect to a basis B of f is produced. Using this set of models Γ, model-based reasoning, via Theorem 6.2, gives the result for relevant queries. Also, given basis B, the algorithm developed in [Bsh93] produces as a byproduct the set of minimal models of f with respect to B, provided that f has small DNF. Combining this with Theorem 6.4 we get the result for common queries. ■

The above theorem is an example for a reasoning problem that is provably hard in the "traditional" sense and has an efficient solution in the new model. Given a CNF knowledge base, even with the added information that it has a short DNF, the reasoning problem is still hard. This is so since it is NP-hard to find a satisfying assignment for a CNF expression even if one knows that it has exactly one satisfying assignment [VV86]. In this case, the additional reasoning power of the agent is gained through the interaction with the world by using $EQ(f)$.

6.5 Learning to Reason without Learning to Classify

In this section we present two results on Learning to Reason any Boolean function f with a polynomial size DNF. These results are significant since there is no known algorithm that Learns to Classify DNF. We present two algorithms. The first algorithm (Theorem 6.6) makes use of a Reasoning Query Oracle $RQ(f, \mathcal{Q})$ and a Membership Query Oracle $MQ(f)$ to *exactly* Learn to Reason for the problem $(f, \mathcal{Q})$. It is worth noticing that this algorithm exhibits an Exact-Learning to Reason algorithm even though its knowledge base consists of an approximate description of the world. The second, more complicated algorithm, uses a "weaker" interface, $EX_D(f)$ instead of $RQ(f, \mathcal{Q})$, and yields PAC-Learn to Reason algorithm for f. The main idea in both algorithms is that it is sufficient to learn the least upper bound of a function in order to reason with common queries.

It is interesting to note that in some sense, a PAC-Learn to Reason algorithm for any Boolean function is implied from the one-time sampling approach developed in Section 3. However, the version presented here has two additional properties. First, the sampling complexity of the algorithm in Theorem 6.8 depends on the size of the class $\mathcal{F}$ which represents the world, as opposed to the one-time sampling, where it depends on the size of the class $\mathcal{Q}$, from which the queries are taken. Second, and more significant, is the fact that the algorithm in Theorem 6.8, while guaranteeing only PAC behavior, has the property that with a bounded number of reasoning mistakes it converges to yield exact reasoning performance. We believe that this property is useful for a Learning to Reason algorithm.

We now present the algorithms. Both are based on a modified version of Bshouty's algorithm to learn Boolean functions via the monotone theory [Bsh93].

Algorithm Ex-L2R-DNF

1. $\forall b \in B$, initialize $\Gamma_b \leftarrow \emptyset$.
2. $\alpha \leftarrow RQ(f, \mathcal{Q})$
3. Answer $f \models \alpha$ by performing model-based test on $\Gamma = \cup_{b \in B} \Gamma_b$.
4. If "wrong" then
5. let x be the counterexample received from $RQ(f, Q)$
6. $\forall b \in B$ such that $x \notin \mathcal{M}_b(\Gamma_b)$
7. $\Gamma_b \leftarrow \Gamma_b \cup$ Find-min-model(x,b)
8. GoTo 2

Figure 1: The Algorithm *Ex-L2R-DNF*

Procedure Find-min-model(x,b)

1. If $\exists y \in \lfloor x \rfloor_b$ such that $f(y) = 1$ /* use MQ(f) */
2. $x \leftarrow y$; GoTo 1
3. Else
4. Return(x)

Figure 2: The Procedure *Find-min-model(x,b)*

6.5.1 Exact Learning to Reason

Let B be a basis for the class of queries $\mathcal{Q}$. The algorithm collects a set of models $\Gamma = \cup_{b \in B} \Gamma_b$, the set of *locally* minimal assignments of f with respect to B. Since this set contains the set of minimal models of f_{lub} with respect to the basis B it guarantees, by Theorem 6.4, exact reasoning with respect to queries in $\mathcal{Q}$. The algorithm *EX-L2R-DNF* is depicted in Figure 1. The following theorem proves its correctness.

Theorem 6.6 *Algorithm Ex-L2R-DNF is a MB-Learn to Reason algorithm for the problem* $(DNF, \mathcal{Q})$*, where* $\mathcal{Q}$ *is the class of all common queries.*

Proof: Denote $h = \wedge_{b \in B} (\vee_{z \in \Gamma_b} \mathcal{M}_b(z))$. Clearly, at every step in the algorithm, $\Gamma \subseteq f$, and therefore the algorithm *Ex-L2R-DNF* never makes a mistake when it says "no" (and is therefore well defined). Whenever the algorithm errs on an $RQ(f, \mathcal{Q})$ query α, it receives from the oracle a positive counterexample, $x \in f \setminus \alpha$. We first argue that $x \in f \setminus h$. In order to show that, we prove that $h \subseteq \alpha$. Indeed, let y be such that $h(y) = 1$. Then, $\forall b \in B$, $\exists z_b \in \Gamma_b$ such that $z_b \leq_b y$. Since the response on the reasoning query was "yes", we have that in particular $\alpha(z_b) = 1$, for all those points z_b. Now, since $\alpha \in \mathcal{Q}$, using Corollary 6.1 we get that $\alpha(y) = 1$.

Now, since $x \notin h$, it is negative for at least one of the b's in the conjunction defining h. Therefore, there exists a model $z \in \min_b(f) \setminus \Gamma_b$ for each such b. Therefore, in this case, the algorithm can use

Algorithm PAC-L2R-DNF

Learning Phase:

1. $\forall b \in B$, initialize $\Gamma_b \leftarrow \emptyset$.
2. Let $h = \wedge_{b \in B} (\vee_{z \in \Gamma_b} \mathcal{M}_b(z))$.
3. Call $EX_D(f)$ $m = \frac{1}{\epsilon} \log \frac{1}{\delta}$ times. Let S be the set of all m samples.
4. If $\nexists x \in S$ such that $f(x) = 1$ and $h(x) = 0$ GoTo 9 /* no positive counterexample */
5. $\forall x \in S$ such that $f(x) = 1$ and $h(x) = 0$ do: /* positive counterexample */
6. $\forall b \in B$ such that $x \notin \mathcal{M}_b(\Gamma_b)$ do:
7. $\Gamma_b \leftarrow \Gamma_b \cup$ Find-min-model(x,b)
8. GoTo 2
9. Return $\Gamma = \cup_{b \in B} \Gamma_b$

Reasoning Phase:

Answer queries by performing model-based reasoning on $\Gamma = \cup_{b \in B} \Gamma_b$

Figure 3: The Algorithm *PAC-L2R-DNF*

the procedure Find-min-model(x,b) to find a new model of f, an element of $min_b^*(f)$. Find-min-model(x,b) is a standard procedure ([Ang88, Bsh93]) that uses a sequence of calls to $MQ(f)$ to find an element of $min_b^*(f)$. Since for all $b \in B$, $|min_b^*(f)| \leq |DNF(f)|$, the size of this representation is polynomial. Moreover, $\Gamma_f^B \subseteq \cup_{b \in B} min_b^*(f) \subseteq f$.

The algorithm might make reasoning mistakes as long as there is an element of Γ_f^B missing from its model based representation, but with every such mistake it makes progress toward collecting the elements in the set Γ_f^B. Therefore, after at most $|\cup_{b \in B} min_b^*(f)| \leq |B| \cdot |DNF(f)|$ mistakes algorithm *Ex-L2R-DNF* makes no more mistakes on $RQ(f, \mathcal{Q})$ queries and by Theorem 6.3, $h = f_{lub}$.

Therefore, Theorem 6.4 implies that *Ex-L2R-DNF* guarantees exact reasoning on queries from $\mathcal{Q}$. ■

6.5.2 PAC Learning to Reason

The algorithm *PAC-L2R-DNF* collects a set of locally minimal models of f with respect to a fixed basis B, a basis for $\mathcal{Q}$. By Theorem 6.3 this yields a representation of the least upper bound of f in $\mathcal{Q}$, and by Theorem 6.4, this is enough to solve the $(\mathcal{F}, \mathcal{Q})$ reasoning problem. Unlike the exact L2R algorithm, *EX-L2R-DNF*, here we simulate the $RQ(f, \mathcal{Q})$ oracle by calling the Example Oracle $EX_D(f)$ sufficiently many times. This simulation builds on the standard simulation of Equivalence Query Oracle by calls to Example Oracle ([Ang88]) and uses some nice properties of the reasoning with models framework. On the other hand, this algorithm is not a mistake-bound algorithm, and the time it takes to achieve its final state does not depend on the behavior of the oracles and can be bounded.

Figure 3 describes the algorithm *PAC-L2R-DNF*. For all $b \in B$ we initialize $\Gamma_b = \emptyset$ and maintain

and hypothesis $h = \wedge_{b \in B} (\vee_{z \in \Gamma_b} \mathcal{M}_b(z))$. To get counterexamples, we simulate $EQ(f)$ by $m = \frac{1}{\epsilon} \log \frac{1}{\delta}$ calls to $EX_D(f)$. On a positive counterexample the algorithm looks for a *locally* minimal assignment of f to be added to Γ_b. Such an assignment always exists since this example is negative for at least one of the b's in the intersection of h. Finding the locally minimal assignment is done by calling the procedure *Find-min-model(x,b)* that uses $MQ(f)$ greedily to find a locally minimal element. That is, the algorithm increases the size of at least one of the Γ_b's. Whenever the algorithm receives a negative counterexample, it simply ignores it. The algorithm stops if m consecutive calls to $EX_D(f)$ do not return any positive counterexample.

We note that simulating $EQ(f)$ by $EX_D(f)$ is essential to our algorithm. Otherwise, an adversarial equivalence oracle can adapt its strategy to the hypothesis that the algorithm holds. In particular, it could prevent the algorithm from making progress by presenting the same negative counterexample forever. The following lemma shows that with an example oracle this is not the case.

Lemma 6.7 *Let f be a Boolean function with a polynomial size DNF representation and let f_{lub} be its least upper bound in $\mathcal{Q}$. Then algorithm PAC-L2R-DNF runs in time polynomial in n and, with probability $> 1 - \delta$ reaches the reasoning phase with a hypothesis h which has the following properties: (1) $h \subseteq f_{lub}$ (2) $Prob_D[f \setminus h] < \epsilon$.*

Proof: Property (1) easily follows from Theorem 6.3 and from the fact that at any point the algorithm holds in Γ_b only (locally) minimal assignment of f.

For (2), we first note that the size of Γ_b for any basis assignment is bounded by the DNF size of f, and is therefore polynomial. This implies that there is a polynomial bound on the number of positive counterexamples that the algorithm might receive. Now consider the algorithm's hypothesis when it stopped. If the algorithm already used up its mistake bound then by Theorem 6.3, $h = f_{lub}$ and we are done. Otherwise, it stopped because it did not receive a positive counterexample from the simulation of the equivalence query. By the selection of m, this implies that with probability $1 - \delta$, $Prob_D[f \setminus h] < \epsilon$. ■

The following theorem shows that the properties of Lemma 6.7 guarantee the correctness of the algorithm.

Theorem 6.8 *Algorithm PAC-L2R-DNF is a PAC-Learn to Reason algorithm for the problem $(DNF, \mathcal{Q})$, where $\mathcal{Q}$ is the class of all common queries.*

Proof: Let $\Gamma = \cup_{b \in B} \Gamma_b$ be the output of the learning phase of the algorithm. Γ is a subset of the set of all locally minimal assignments of f with respect to B.

To prove the correctness of the algorithm consider first the case in which $f \models \alpha$. Since $\Gamma \subseteq f$, $\forall x \in \Gamma, \alpha(x) = 1$ and the algorithm answers "yes".

Assume now that $f \not\models \alpha$, and suppose, by way of contradiction, that algorithm answers "yes" on α. Since α is (f, ϵ)-fair, we know that $Prob_D[f \setminus \alpha] > \epsilon$. Since by Lemma 6.7, $Prob_D[f \setminus h] < \epsilon$, this implies that $h \setminus \alpha \neq \emptyset$.

Let $y \in h \setminus \alpha$. Since, by the assumption, the algorithm answers "yes", we know that for all $u \in \Gamma$, $\alpha(u) = 1$. Since B is a basis for α and for all $u \in \Gamma$, $\alpha(u) = 1$, Corollary 6.1 implies that

$$\forall u \in \Gamma, \ \forall b \in B, \ \exists v_{u,b} \in \min_b(\alpha), \text{ such that } u \geq_b v_{u,b}. \tag{7}$$

The fact $y \in h = \wedge_{b \in B} (\vee_{z \in \Gamma_b} \mathcal{M}_b(z))$, we have that

$$\forall b \in B, \ \exists z \in \Gamma_b, \text{ such that } y \geq_b z. \tag{8}$$

By the assumption, all the elements z identified in Equation 8 satisfy α and therefore, as in Equation 7 we have that

$$\forall z \in \Gamma,\ \forall b \in B,\ \exists v_{z,b} \in \min_b(\alpha) \text{ such that } z \geq_b v_{z,b}. \tag{9}$$

Substituting Equation 9 into Equation 8 gives

$$\forall b \in B, \exists v_{(z),b} \in \min_b(\alpha) \text{ such that } y \geq_b v_{(z),b}$$

which implies, by Corollary 6.1, that $\alpha(y) = 1$, a contradiction. ■

We have shown that there exist Learning to Reason algorithms for the all Boolean functions with a polynomial size DNF, provided that the queries are common. This should be contrasted with the inability to *learn to classify* DNF. One can learn f_{lub} and reason with it with respect to common queries, but f_{lub} is not sufficient as a substitute for f when classifying examples, since we cannot bound the size of $Prob_D[h \setminus f]$.

6.6 Structure Identification

We briefly describe another application of Model Based Reasoning.

Dechter and Pearl investigate in [DP92] the problem of identifying tractable classes of CNF formulas. In particular they consider the following problem: Given a set ρ of models, can one:

1. Find a Horn theory f such that $f = \rho$, if one exists.
2. Find a Horn theory f such that $\rho \subseteq f$ and there is no Horn function g such that $\rho \subseteq g \subseteq f$.

In [DP92] it is shown ("Horn theories are identifiable", Corollary 4.11) that when ρ is a set of models of a Horn theory, a theory f can be found in time polynomial in $|\rho|$. The second problem ("strong-identifiability of Horn-theories"), which is just the problem of finding ρ^H_{lub}, (i.e., the least upper bound of ρ with respect to Horn) is left open.

The interest in the question of the identifiability and strong identifiability of Horn theories is motivated by the fact that Horn theories are tractable, and in particular, given a CNF formula in a Horn form, reasoning with it can be performed efficiently. Therefore, the significance of the strong-identifiability is that given a set ρ of models, one could perform efficient reasoning with a theory that is the best Horn approximation of ρ. We show now that the techniques developed in this paper solve that above problem (without having to find a Horn formula).

Claim 6.9 *Let ρ be a set of models and g the least Horn upper bound of ρ. Then,*
(i) A closed form formula for the required approximation can be written as:

$$g = \wedge_{B_H} \vee_{z \in \rho} \mathcal{M}_b z. \tag{10}$$

(ii) For every query α, the reasoning problem $g \models \alpha$ can be answered correctly using the set ρ.

Proof: (i) is immediate from Theorem 6.3, since $g = \rho^H_{lub}$. For (ii) notice that if the query α is a Horn query, then model based reasoning with ρ is correct, by Theorem 6.4. For correct reasoning with general queries, it is possible to use the reasoning algorithm from [KKS93]. ■

7 Learning to Reason with Horn Queries

In previous sections (e.g, Theorem 6.5, Theorem 6.6) we have shown that for various worlds we can learn to reason with respect to all common queries. We give here an orthogonal result that shows that we can learn to reason from every Boolean function (with polynomial size Horn-least upper bound), with respect to Horn queries. The results is based of an algorithm developed by Frazier and Pitt [FP93] for the purpose of learning to classify Horn theories.

Another difference from the previous section is that the oracles used here, following [FP93], are entailment oracles. The agent "learns" the world from examples that are statements that the world entails and does not entail, rather then, say, seeing positive and negative instances or the world, as in $EX_D(f)$ and $MQ(f)$. The following definitions are taken from [FP93]:

Definition 7.1 *An* Entailment Membership Query Oracle *for a function f, denoted $EnMQ(f)$, is an oracle that when given as input a function g answers "yes" if $f \models g$ and "no" otherwise.*

Definition 7.2 *An* Entailment Equivalence Query Oracle *for a function f, denoted $EnEQ(f)$, is an oracle that when given as input a function g, answers "yes" if and only if $f \equiv g$. If it answers "no" it supplies either a negative counterexample, namely, a function h such that $f \models h$ but $g \models h$, or a positive counterexample h such that $f \models h$ but $g \not\models h$. The function h is restricted to be a Horn disjunction.*

Using the above oracles, it can be shown that the algorithm LRN presented in [FP93] can be used to find the least upper bound of any Boolean function f with respect to Horn and thus, by Theorems 6.3 and 6.4, this can be used further to reason with Horn queries.

In particular, this provides another example in which there exists an efficient L2R algorithm but reasoning is hard, similar to Section 6.4. In this case, the problem $f \models \alpha$, even when α is a Horn query, is in general hard if f is given as a CNF formula. We summarize in the next theorem:

Theorem 7.1 *There is an Exact-Learn to Reason algorithm, that uses an Entailment Equivalence Query oracle and an Entailment Membership Query Oracle, for $(\mathcal{F}, H)$, where $\mathcal{F}$ is any class of all Boolean functions with polynomial size Horn least upper bound.*

8 Conclusions

We have introduced a new framework for the study of reasoning in intelligent systems. The approach suggested, *Learning to Reason*, is intended to overcome some of the basic problems in the traditional approach to reasoning. This framework differs from existing ones in that it sees learning as an integral part of the process, it avoids the rigid syntactic restrictions on the intermediate knowledge representation, and it makes explicit the dependence of the reasoning performance on the input from the environment.

We have shown that under some restrictions on the queries asked (which depend on the "world"), Learning to Reason is possible for all propositional languages. This should be contrasted with the hardness of traditional reasoning even with very restricted languages.

We discussed the relation of the new framework to the known computational approaches to learning and to reasoning, and have shown how previous results in learning and reasoning can be used in the new framework.

Further, we exhibited cases in which the new framework allows for successful algorithms, but stated separately, either the reasoning problem or the learning problem are not (or not known to be) solvable.

We believe that this framework is a step toward constructing an adequate computational theory of reasoning. In particular, it draws attention to many issues that should be addressed, we believe, in developing such a theory. We mention just two such issues that do not get much attention in the traditional reasoning framework. The first is quantifying the reasoning performance relative to the "world" and to a restricted set of queries. The second is the consideration of the plausible interface between the reasoner and the world (instead of, or in addition to the traditional formula-based representation of a theory).

Acknowledgments

We are grateful to Les Valiant for many enjoyable discussions that helped us develop the ideas presented here.

References

[AL88] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[Ams88] J. Amsterdam. Extending the valiant learning model. In *Proceeding of the Fifth International Workshop on Machine Learning*, pages 364–375, June 1988.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[Ang92] D. Angluin. Computational learning theory: Survey and selected bibliography. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, pages 351–369, May 1992.

[AS91] D. Angluin and D. K. Slonim. Learning monotone DNF with an incomplete membership oracle. In *Proceedings of COLT '91*, pages 139–146. Morgan Kaufmann, 1991. (To appear in *Machine Learning.*).

[BL84] R. Brachman and H. Levesque. The tractability of subsumption in framebased description languages. In *Proceedings of the National Conference on Artificial Intelligence*, pages 34–37, 1984.

[Bro91] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Bsh93] N. H. Bshouty. Exact learning via the monotone theory. In *Proceedings of the IEEE Symp. on Foundation of Computer Science*, pages 302–311, Palo Alto, CA., 1993.

[DP91] J. Doyle and R. Patil. Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.

[DP92] R. Dechter and J. Pearl. Structure identification in relational data. *Artificial Intelligence*, 58:237–270, 1992.

[FP93] M. Frazier and L. Pitt. Learning from entailment: An application to propositional Horn sentences. In *Proceedings of the International Conference on Machine Learning*. Morgan Kaufmann, 1993.

[Hau87] D. Haussler. Bias, version spaces and Valiant's learning framework. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 324–336, University of California, Irvine, June 1987.

[Kea92] M. Kearns. Oblivious pac learning of concepts hierarchies. In *Proceedings of the National Conference on Artificial Intelligence*, pages 215–222, 1992.

[Kir91] D. Kirsh. Foundations of AI: the big issues. *Artificial Intelligence*, 47:3–30, 1991.

[KKS93] H. Kautz, M. Kearns, and B. Selman. Reasoning with characteristic models. In *Proceedings of the National Conference on Artificial Intelligence*, pages 34–39, 1993.

[KL88] M. Kearns and M. Li. Learning in the presence of malicious errors. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 267–280, Chicago, Illinois, May 1988.

[KR94a] R. Khardon and D. Roth. Learning to reason with a restricted world view. 1994. in preparation.

[KR94b] R. Khardon and D. Roth. On defaults and relevance in model-based reasoning. In *AAAI Fall Symposium on Relevance*, 1994. Forthcoming.

[KR94c] R. Khardon and D. Roth. Reasoning with models. In *Proceedings of the National Conference on Artificial Intelligence*, pages xx–yy, 1994. Full version: Technical Report TR-1-94, Aiken Computation Lab., Harvard University, January 1994.

[Lev92] H. Levesque. Is reasoning too hard ? In *Proceeding of the 3rd NEC research Symposium.* 1992.

[Lit89] N. Littlestone. *Mistake bounds and logarithmic linear-threshold learning algorithms.* PhD thesis, U. C. Santa Cruz, March 1989.

[McC58] J. McCarthy. Programs with common sense. In R. Brachman and H. Levesque, editors, *Readings in Knowledge Representation, 1985.* Morgan-Kaufmann, 1958.

[Pap91] C. H. Papadimitriou. On selecting a satisfying truth assignment. In *Proc. 32nd Ann. IEEE Symp. on Foundations of Computer Science*, pages 163–169, 1991.

[Rot93] D. Roth. On the hardness of approximate reasoning. In *Proceedings of the International Joint Conference of Artificial Intelligence*, pages 613–618, August 1993.

[Sel90] B. Selman. *Tractable Default Reasoning.* PhD thesis, Department of Computer Science, University of Toronto, 1990.

[Sha93] L. Shastri. A computational model of tractable reasoning - taking inspiration from cognition. In *Proceedings of the International Joint Conference of Artificial Intelligence*, pages 202–207, August 1993.

[Val84] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

[Val85] L. G. Valiant. Learning disjunctions of conjunctions. In *Proceedings of the International Joint Conference of Artificial Intelligence*, pages 560–566. Morgan Kaufmann, 1985.

[Val94] L. G. Valiant. *Circuits of the Mind*. Oxford University Press, 1994. Forthcoming.

[VV86] L. G. Valiant and V. V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.