

Learning to Reason with a Restricted View*

Roni Khardon[†]

Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ
Scotland
roni@dcs.ed.ac.uk

Dan Roth[‡]

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Ave.
Urbana, IL 61801
USA
danr@cs.uiuc.edu

Abstract

The Learning to Reason framework combines the study of Learning and Reasoning into a single task. Within it, learning is done specifically for the purpose of reasoning with the learned knowledge. Computational considerations show that this is a useful paradigm; in some cases learning and reasoning problems that are intractable when studied separately become tractable when performed as a task of Learning to Reason.

In this paper we study Learning to Reason problems where the interaction with the world supplies the learner only partial information in the form of partial assignments. Several natural interpretations of partial assignments are considered and learning and reasoning algorithms using these are developed. The results presented exhibit a tradeoff between learnability, the strength of the oracles used in the interface, and the range of reasoning queries the learner is guaranteed to answer correctly.

1 Introduction

Most of the work in computational learning theory is concerned with the problem of concept learning; the performance of the learner is measured by how well it classifies future examples, according to whether they belong to a target concept or not. We therefore call this the task of *Learning to Classify*. In contrast, the *Learning to Reason* framework studied in this paper is aimed at situations where the performance is measured by the learner's ability to reason about its environment. In some sense, within this framework, the learner acquires "domain knowledge"

* An earlier version of the paper appears in the Proceedings of the Workshop on Computational Learning Theory, COLT-95.

[†]Most of this work was done while the author was at Harvard University and supported by ARO grant DAAL03-92-G-0115 and ONR grant N00014-95-1-0550.

[‡]Most of this work was done while the author was at Harvard University supported by NSF grant CCR-92-00884, by DARPA AFOSR-F4962-92-J-0466 and by ONR grant N00014-96-1-0550.

that, in turn, is used for reasoning about the domain. This framework, introduced in (Kharon and Roth, 1994a), follows standard models of learning (Valiant, 1984; Angluin, 1988) in the way it models the interaction of the learner with its environment in the process of learning. On the other hand, it highlights the fact that new learning questions arise and should be addressed if we want to learn in order to reason.

We consider cases where the domain can be captured by a propositional expression W . The reasoning task is that of deduction; namely, deciding whether $W \models \alpha$ where α is a propositional expression capturing a question presented to it. By interacting with its interface, the learner constructs some representation KB of the information it has learned, and uses it to answer reasoning questions. Namely, for a question α presented to it, the learner must answer correctly whether $W \models \alpha$ or $W \not\models \alpha$ by using its representation KB .

It was shown in (Kharon and Roth, 1994a) that the Learning to Reason framework offers efficient solutions in cases where the separate learning and reasoning problems are intractable. Such results are made possible by a judicious choice of restrictions on the language of queries α the learner is required to answer correctly, and the use of knowledge representations that facilitate efficient reasoning. Thus, the framework can help in concentrating the effort towards finding representations that are useful both for learning and for reasoning. Moreover, the results suggest that restrictions leading to efficient solutions are best sought in the combined framework.

In this paper we study Learning to Reason problems in which the interaction with the environment is via partial observations. This is modeled by assigning some of the learner’s measurements the value “unknown”. We discuss a few ways in which partial examples can be interpreted, including cases where unobserved attributes are irrelevant, and cases where unobserved attributes are arbitrarily (and possibly adversarially) hidden from the learner. The goal of the learner is to reason about this partially observable world by answering reasoning questions. As in (Kharon and Roth, 1994a), it is shown that in some cases Learning to Reason can be achieved even though the traditionally phrased reasoning problem is intractable, and when the traditionally phrased learning problem – learning a representation of the world (a concept) – is intractable.

Our positive results use a model based approach to reasoning (Kautz, Kearns, and Selman, 1995; Kharon and Roth, 1994b). In this approach, the agent keeps a collection of (partial) assignments as a knowledge base, and the reasoning task is performed simply by evaluating the queries on this set of models. The need to study these knowledge representations arises from the intractability of reasoning with the standard formula-based representations. In particular, we show that Learning to Reason (L2R) is possible if the learner has access to random partial assignments, similar to examples in the PAC-learning model, under some probabilistic assumption on the queries. We also show that on-line L2R algorithms exist for queries in k -CNF. Finally, we show that with stronger oracles, either a membership oracle (Angluin, 1988) for partial observations or entailment queries (Frazier and Pitt, 1993), L2R is possible for the same classes of queries discussed in (Kharon and Roth, 1994a). In fact, our proofs show that these oracles are strong enough to enable the learner to complete partial observations into total ones and learn in this way.

Our framework is related to work in Inductive Logic Programming (ILP) (Muggleton and De Raedt, 1994) where examples for learning are often given in the form of clauses implied by the learned program. In this sense, hence, ILP algorithms use a fixed knowledge representation to learn to reason about such clauses. Some of our results can therefore be phrased in terms of ILP. Using terminology from (De Raedt, 1997) these imply for example that k -CNF is learnable from entailment.

While in this paper we concentrate on deductive reasoning, the learning to reason approach should be seen in a more general context and can be applied for a variety of tasks. In particular, a

different treatment of partial information is taken in (Valiant, 1995; Roth, 1995), where the effect of partially specified *queries* is discussed. A similar learning to reason approach is developed there, supporting several aspects of non-monotonic reasoning (Reiter, 1987) which have proved difficult to capture in other frameworks. The Learning to Reason approach has also been extended to consider several other tasks, including some forms of default reasoning, learning in order to act in the world, and learning of active classifiers (Khardon and Roth, 1997; Khardon, 1996; Greiner, Grove, and Roth, 1996).

The rest of the paper is organized as follows. We start by presenting some preliminaries on reasoning in Section 2 and then introduce our notion of partial observations in Section 3. Section 4 develops ideas on knowledge representations and reasoning procedures that cope with partial observations. Section 5 formally defines the Learning to Reason model, and Section 6 presents the Learning to Reason algorithms. We conclude in Section 7 with a summary.

2 Reasoning

We consider problems of reasoning where the “world” (the domain in question) is modeled as a Boolean function¹ $W : \{0, 1\}^n \rightarrow \{0, 1\}$. Similarly, the knowledge base KB consists of some representation for a Boolean function. Note that we make a distinction between KB and W : KB is the representation used by the algorithm, whereas the reasoning performance is measured relative to W .

Let f, g be Boolean functions. An assignment $x \in \{0, 1\}^n$ is a *model* (satisfying assignment) of f if $f(x) = 1$. By “ f entails g ”, denoted $f \models g$, we mean that every model of f is also a model of g . We also refer to the connective \models by its equivalent, proof theoretic name, “implies”. Since “entailment” and “logical implication” are equivalent, we can treat f either as a Boolean function (usually, using a propositional expression that represents the function), or as 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 $f \subseteq g$. Let \mathcal{F}, \mathcal{Q} be two propositional languages. For every $f \in \mathcal{F}$ the size of f is the size of its representation as an expression in \mathcal{F} . The size for expressions $g \in \mathcal{Q}$ is defined likewise. For DNF and CNF expressions discussed later, the size is respectively, the number of terms or the number of clauses in the representation.

Definition 2.1 *An algorithm A is an efficient and exact reasoning algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q})$, if for all $f \in \mathcal{F}$ and $\alpha \in \mathcal{Q}$, 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 $f \models \alpha$.*

Answering the question $f \models \alpha$ is equivalent to solving unsatisfiability for the formula $f \wedge \bar{\alpha}$. Thus, when f is given in CNF, exact reasoning can be done efficiently only when this satisfiability problem can be solved efficiently (e.g., Horn expressions). We note that if f is given in DNF then reasoning can be done efficiently (for α in CNF), but this representation has been less favored mainly for comprehensibility reasons. (Since a representation as a set of rules easily translates to a CNF but not to a DNF expression.) The following class of queries plays an important role in our results:

¹This is equivalent to the definition in terms of propositional expressions. A propositional expression is just a representation for a Boolean function, and a propositional language is a class of representations for Boolean functions. These terms are used in the reasoning and learning literature respectively, and we use them interchangeably.

Definition 2.2 *The class \mathcal{Q}_C of common queries consists of Boolean functions with the following property: Every $\alpha \in \mathcal{Q}_C$ has a CNF representation, in which every clause is either (1) of size $\leq \log n$ or (2) a Horn-clause (contains at most one positive literal) or (3) a k -quasi-Horn clause (contains at most k positive literals).*

3 Partial Observations

We assume that the learner interacts with the world via a set of measurements it makes and predicates it computes from these measurements. The measurements are abstracted as a set of binary attributes modeled by the set $X = \{x_1, \dots, x_n\}$ of variables. Each of these is associated with an attribute of the world and can take the values 1 or 0 to indicate whether the associated attribute is true or false in the world. While at each point in time the agent can take measurements, in general it would not be possible to observe all the attributes at all times. Thus, while some attributes may be known to be true or false others may not be observed. We denote this situation by assigning the value $*$ to such attributes, so that the input an agent sees is a *partial assignment* to the n variables – an assignment in $\{0, 1, *\}^n$. For example, $v = (1 * 0)$ means that x_1 is true, x_3 is false, and the value of x_2 is unknown. An assignment is *total* if the value of every variable is *known* (i.e., assigned value from $\{0, 1\}$). An assignment y is an *extension* of x if y agrees with x on all the variables assigned 0 or 1 in x (and where variables assigned $*$ in x may be assigned 0 or 1 in y).

We start by discussing several interpretations for the information conveyed by partial observations. Recall that we model the world as a Boolean function $W : \{0, 1\}^n \rightarrow \{0, 1\}$, where the intention is that $W(x) = 1$ if and only if x corresponds to a combination of features which is possible in the world.²

There are various ways to interpret the meaning that an observation $v \in \{0, 1, *\}^n$ conveys on W . In particular:

1. **Universal interpretation:** For all possible extensions of v to total models v' , $W(v') = 1$.
2. **Existential interpretation:** There exists an extension of v to a total model v' , such that $W(v') = 1$.
3. **Abbreviated interpretation:** All variables assigned $*$ in v are assumed to have the value 0. That is, v corresponds to v' where $v'_i = v_i$, for all i such that $v_i = 1$, and $v'_i = 0$, otherwise.

In the following we assume that one of the above interpretations has been chosen, and is fixed for the entire duration of the learning scenario. In such a case we can evaluate a Boolean function on a partial assignment according to this interpretation. We denote this by using a subscript to mark the interpretation chosen. Namely, f_u (f_e , f_a) means that the function f is evaluated according to the Universal (Existential, Abbreviated) interpretation.

Several works have dealt with partial observations in the context of concept learning. The approach (1) is taken by Valiant (1984) to model an agent that observes all the attributes that are relevant for the classification of the learned concept. The approach (3), can be thought of as a “closed world assumption” on attribute values. It is useful when the total number of attributes

²Clearly, combinations of features that cannot happen in the world cannot normally be observed through the sensors of a real agent. However, one can imagine a teacher or an active learner generating negative examples. In some situations, though, it may make sense to restrict the observations to positive ones. The framework studies the general case and results for restricted cases can be derived from ones presented here.

n is much larger than the number of positive attributes any one example has, and an example is presented as a list of its positive attributes. Concept learning in this model is studied in (Blum, 1992; Blum, Hellerstein, and Littlestone, 1991). For the task of learning a “world” representation in order to reason about it later, it seems that the agnostic approach, the existential interpretation (2), ought to be taken.

Several other works (Ben-David and Dichterman, 1993; Greiner, Grove, and Kogan, 1996; Schurmans and Greiner, 1994) have studied partial assignments in different settings, mostly for learning to classify, and the models and results are incomparable with ours. The work in (Valiant, 1995; Roth, 1995) is closer in that it handles reasoning tasks. In contrast with the above interpretations they treat the unobserved value $*$ as a third valid value. This is used to show that several non-monotonic reasoning phenomena can be explained through learning.

3.1 Reasoning in the Presence of Partial Observations

We extend the notion of models and implication for partial assignments. This is done in order to facilitate the use of partial assignments by reasoning algorithms in the next section. The definition of a model can be extended to $\{0, 1, *\}^n$, using the interpretations given above. A partial assignment $x \in \{0, 1, *\}^n$ is a p -model of W if and only if $W_p(x) = 1$, for $p \in \{a, e, u\}$, depending on the interpretation we favor. Likewise we define implication with respect to partial assignments: Let α be a Boolean function. We say that W p -implies α ($W \models_p \alpha$) if every p -model of W is also a p -model of α (where $p \in \{a, e, u\}$). As the following theorem shows the connectives \models and \models_p are equivalent. This is implicitly used later by reasoning algorithms that upon finding $v \in \{0, 1\}^n$ such that $W_p(v) = 1$ and $\alpha_p(v) = 0$ conclude that $W \not\models \alpha$.

Theorem 3.1 *Let W, α be Boolean functions and $p \in \{a, e, u\}$. Then, $W \models \alpha$ if and only if $W \models_p \alpha$.*

Proof: Assume first that $W \models \alpha$. Let $x \in \{0, 1, *\}^n$ such that $W_e(x) = 1$. Then, there exists an extension $x' \in \{0, 1\}^n$ of x such that $W(x') = 1$. From the assumption, $\alpha(x') = 1$ and therefore $\alpha_e(x) = 1$. Similarly, let $x \in \{0, 1, *\}^n$ such that $W_u(x) = 1$. Then, for all extensions $x' \in \{0, 1\}^n$ of x , $W(x') = 1$. Therefore, all those extensions satisfy $\alpha(x') = 1$ and we have that $\alpha_u(x) = 1$. Finally, let $x \in \{0, 1, *\}^n$ such that $W_a(x) = 1$. Then, the unique 0-padded extension $x' \in \{0, 1\}^n$ of x satisfies $W(x') = 1$. From the assumption, $\alpha(x') = 1$ and therefore $\alpha_a(x) = 1$. We have shown that $W \models_p \alpha$ for all $p \in \{a, e, u\}$.

For the other direction, assume first that $W \models_e \alpha$. Then, given $x \in \{0, 1\}^n$ such that $W(x) = 1$, we can treat x as an element of $\{0, 1, *\}^n$ and deduce, from the assumption, that $\alpha(x) = 1$. Therefore, $W \models \alpha$. The same argument holds when we assume that $W \models_u \alpha$. Assume now that $W \models_a \alpha$. Then, given $x \in \{0, 1\}^n$ such that $W(x) = 1$, we define $x' \in \{0, 1, *\}^n$ by replacing all the 0 entries in x by $*$'s. By definition, $W_a(x') = 1$ and therefore $\alpha_a(x') = 1$, and we get that $\alpha(x) = 1$, that is, $W \models \alpha$. ■

3.2 Computational Considerations

While the evaluation of Boolean functions is trivial for total models, model evaluation may be hard for partial assignments. Consider for example $y = (1 * 0)$ and the term $t_y = x_1 \wedge \overline{x_3}$, and clause $c_y = \overline{x_1} \vee x_3$ that can be constructed from it. By definition, for any f we have that $f_u(y) = 1$ if and only if all extensions of y satisfy f . In other words $f_u(y) = 1$ if and only if $t_y \models f$. Similarly, it can be seen that $f_e(y) = 0$ if and only if $f \models c_y$. These observations are formalized below.

Definition 3.1 Given a partial assignment $y \in \{0, 1, *\}^n$ define a corresponding term and a corresponding clause

- $t_y = \bigwedge_{i=1}^n x_i^{y_i}$
- $c_y = \overline{t_x} = \bigvee_{i=1}^n x_i^{1-y_i}$

where $x_i^0 = \overline{x_i}$, $x_i^1 = x_i$, $x_i^* = 1$, $x_i^{1-*} = 0$.

Similarly, given a term t or clause c , define corresponding partial assignments $y_t, y_c \in \{0, 1, *\}^n$, such that:

- $(y_t)_j = \begin{cases} 1 & \text{if } x_j \in t \\ 0 & \text{if } \overline{x_j} \in t \\ * & \text{otherwise} \end{cases}$
- $(y_c)_j = \begin{cases} 1 & \text{if } \overline{x_j} \in c \\ 0 & \text{if } x_j \in c \\ * & \text{otherwise} \end{cases}$

where $(y_t)_j, (y_c)_j$ are the j th bits in y_t, y_c , respectively.

Clearly, the definitions imply that $y_{c_y} = y$ and $c_{y_c} = c$ and similarly $y_{t_y} = y$ and $t_{y_t} = t$. The following claim is immediate from the definitions:

Claim 3.2 Let $y \in \{0, 1, *\}^n$ be a partial assignment, t a term, c a clause and $f : \{0, 1\}^n \rightarrow \{0, 1\}$. Then

- (1) $f_u(y) = 1$ if and only if $t_y \models f$.
- (2) $t \models f$ if and only if $f_u(y_t) = 1$.
- (3) $f_e(y) = 0$ if and only if $f \models c_y$.
- (4) $f \models c$ if and only if $f_e(y_c) = 0$.

The above claim shows that evaluating f on a partial assignment may be hard. If we take the universal interpretation, the claim shows that evaluating $f_u(y)$ is equivalent to solving satisfiability for $(t_y \wedge \overline{f})$, and therefore is easy for f that is given in a CNF representation, but co-NP-Complete for f given in a DNF representation. On the other hand, if we take the existential interpretation, evaluating $f_e(y)$ is equivalent to solving satisfiability for $f \wedge \overline{c_y}$, and is therefore easy if f is given in a DNF representation, but when f is given in a CNF representation it is efficient only for some restricted subsets (e.g., Horn CNF formulas, 2-CNF formulas, log n -clause CNF formulas). When using the abbreviated interpretation, evaluating f on a partial assignment is just as easy as evaluating it on a total assignment, the unique total assignment that is equivalent to this partial assignment. Since the common assumption in knowledge representation and reasoning is that queries are represented in a CNF form, when trying to evaluate a query under the existential representation we need to make sure that we restrict ourselves to deal with queries that can be evaluated efficiently. We will show how to get around this problem by using a modified evaluation procedure.

As an aside we mention that the above discussion shows that when dealing with partial assignments, classification problems (e.g., evaluating a given function on an assignment) can be viewed as reasoning problems. Similar observations were recently found useful in the context of Inductive Logic Programming (De Raedt, 1997; Khardon, 1998) where examples may be given in the form of clauses.

Algorithm MBR(Γ, α):

Test Set: A set $\Gamma \subseteq W$ of possible assignments.

Test: If there is an element $x \in \Gamma$ such that $\alpha(x) = 0$, return “No”. Otherwise, return “Yes”.

Figure 1: MBR: Model-Based Reasoning

4 Reasoning with Partial Assignments

In this section we study knowledge representations based on partial assignments. This is done by extending previous results on reasoning with total models.

4.1 Reasoning with Total Models

We have given a model-theoretic definition of the implication relation \models . The algorithm **MBR**, described in Figure 1 uses this definition in a straightforward way: it maintains a set of models $\Gamma \subseteq W \subseteq \{0, 1\}^n$ as its knowledge base. To decide whether $W \models \alpha$ it checks, for all the models $z \in \Gamma$, whether $\alpha(z) = 1$. If for some z , $\alpha(z) = 0$, it says “No”; otherwise it says “Yes”.

By definition, if $\Gamma = W$ this approach yields correct deduction, but representing W by explicitly holding *all* the possible models of W is not plausible. A model based approach becomes feasible if one can make correct inferences when working with a small subset of models. Some results along this line have been obtained (Kautz, Kearns, and Selman, 1995; Khardon and Roth, 1994b). In particular, previous work in (Khardon and Roth, 1994b), using ideas from (Bshouty, 1995), identified a small set of models, called the set of *characteristic models* of W , that supports correct reasoning. We briefly describe some of the relevant results we need, culminating in Theorem 4.1 that identifies the models needed for the algorithm **MBR** to be correct and efficient.

Definition 4.1 (Order) We denote by \leq the usual partial order on $\{0, 1\}^n$, 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 $x \oplus b \leq y \oplus b$, where \oplus denotes the XOR operation (bitwise addition modulo 2). As with other order relations, $x \leq_b y$ can also be written as $y \geq_b x$, and if $x \leq_b y$ and $x \neq y$ we write $x <_b y$.

Intuitively, if $b_i = 0$ then the order relation on the i th bit is the normal order; if $b_i = 1$, the order relation is reversed and we have that $1 <_{b_i} 0$.

Definition 4.2 The monotone extension of f with respect to b is:

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

Definition 4.3 A set B is a basis for f if $f = \bigwedge_{b \in B} \mathcal{M}_b(f)$. B is a basis for a class of functions \mathcal{F} if it is a basis for all the functions in \mathcal{F} .

It is known (Bshouty, 1995; Khardon and Roth, 1994b) that there are classes of functions which have a small basis. In particular, the class of common queries defined above has a polynomial size basis B_c , and the set $B_{H_k} = \{u \in \{0, 1\}^n \mid \text{weight}(u) \geq n - k\}$ is a basis for the class of k -quasi-Horn functions.

Algorithm Lazy-MBR(Γ, α):*Test Set:* A set Γ of partial satisfying assignments.*Test:* Given a CNF query α , if there is an element $x \in \Gamma$ which falsifies one of the clauses in α , deduce that $f \not\models \alpha$; Otherwise, $f \models \alpha$.

Figure 2: Lazy-MBR: Reasoning with Partial Models

Definition 4.4 Let \mathcal{F} be a class of functions, and let B be a set of assignments in $\{0, 1\}^n$. For $W \in \mathcal{F}$ we define the set $\Gamma = \Gamma_W^B$ of characteristic models of W with respect to B to be the set of all minimal assignments of W with respect to the basis B . Formally,

$$\Gamma_W^B = \cup_{b \in B} \{z \in \min_b(W)\},$$

where

$$\min_b(W) = \{z \mid z \in W, \text{ such that } \forall y \in W, z \not\prec_b y\}.$$

Using these definitions we can identify the models that are needed for the algorithm **MBR**:

Theorem 4.1 (Khardon and Roth, 1994b) Let $W \in \mathcal{F}$, $\alpha \in \mathcal{G}$ and let B be a basis for \mathcal{G} . Then $W \models \alpha$ if and only if for every $u \in \Gamma_W^B$, $\alpha(u) = 1$.

4.2 Reasoning with Partial Models

We next consider whether partial assignments can be used in a similar way. We start by characterizing a useful set of partial models which is derived from partial views of characteristic models.

For any fixed k , there are $\binom{n}{k}$ subsets of size k of the n variables. Given an element $x \in \{0, 1\}^n$ and a subset I of k variables, the projection of x on I is the partial model v defined by: $v_i = x_i$, for all $x_i \in I$, and $v_i = *$ otherwise. Throughout this section, fix B to be a basis for a class which includes k -CNF.³ Let $\Gamma = \Gamma_f^B$ be the set of characteristic models. Projecting all the elements of Γ on all subsets of size k we get a set of at most $|\Gamma| \binom{n}{k}$ partial models.

Definition 4.5 Let $\Gamma = \Gamma_f^B$ be the set of characteristic models. Then the set of all projections of elements of Γ on subsets of size k is denoted by $\Gamma_{f|k}^B$.

Figure 2 describes the algorithm **Lazy-MBR** used for reasoning with partial models. The algorithm keeps partial assignments in its knowledge base, and assumes that the queries are given in CNF form. When it receives a CNF query α , the algorithm checks whether one of the partial assignments in its knowledge base falsifies one of the clauses in α . If it finds such a partial model, it says “No” and otherwise it says “Yes”. Note that the algorithm differs from **MBR** in that it avoids testing whether a partial assignment satisfies a CNF expression but instead tests satisfiability of one clause at a time.

Theorem 4.2 The algorithm **Lazy-MBR**, when using the set $\Gamma_{f|k}^B$, is correct for all queries $\alpha \in k$ -CNF.

³The basis B_{H_k} , of size $O(n^k)$ is sufficient, but there is a smaller basis, composed of a (n, k) -universal set (Naor and Naor, 1993).

	x_1x_2	x_1x_3	x_1x_4	x_2x_3	x_2x_4	x_3x_4
$\gamma_1 = 0000$	0 0	0 0	00	0 0	0 0	00
$\gamma_2 = 0101$	01	00	0 1	10	1 1	0 1
$\gamma_3 = 1110$	1 1	1 1	1 0	1 1	10	1 0
$\gamma_4 = 0100$	0 1	0 0	0 0	1 0	1 0	0 0

Figure 3: Projection Example

Proof: Clearly, if $f \models \alpha$, model based reasoning answers correctly. Assume therefore that $f \not\models \alpha$. Theorem 4.1 implies that there exists a total model $z \in \Gamma_f^B$ such that $\alpha(z) = 0$. In particular, since α is a k -CNF, one of its clauses C_z must be falsified by z . That is $C_z(z) = 0$. Now consider the element z' which is the projection of z on the variables in C_z . Clearly z' is in $\Gamma_{f|k}^B$ and $C_z(z') = 0$. So the lazy model based algorithm will answer “No” due to z' , and is therefore correct. ■

While projection on sets of limited size is useful it does not preserve all the information in the characteristic models:

Claim 4.3 *The set Γ_f^B cannot be reconstructed from $\Gamma_{f|k}^B$. Furthermore, $\Gamma_{f|k}^B$ does not retain all the information needed in order to answer general queries supported by Γ_f^B .*

Proof: We prove the claim by exhibiting an example in which it is impossible to decide which of two sets of models created $\Gamma_{f|k}^B$. Let $n = 4$, $k = 2$, $\Gamma^{(3)} = \{\gamma_1, \gamma_2, \gamma_3\}$ and $\Gamma^{(4)} = \Gamma^{(3)} \cup \{\gamma_4\}$, where $\gamma_1 = 0000$, $\gamma_2 = 0101$, $\gamma_3 = 1110$ and $\gamma_4 = 0100$. Figure 3 shows the projections of elements in $\Gamma^{(4)}$ on subsets of size 2. The first three rows in the figure show all the projections of the elements in $\Gamma^{(3)}$. Since $\Gamma_{f|k}^B$ is a set, and all the elements in the forth line already appear in the previous lines, we get that it is impossible to decide whether $\Gamma_{f|k}^B$ is a projection of $\Gamma^{(3)}$ or $\Gamma^{(4)}$. However, these two sets yield different reasoning behavior, when the queries are not taken from 2-CNF.

For example, consider the functions $f = \{\gamma_1\} \cup \{\gamma_2\} \cup \{\gamma_3\}$ and $g = \{\gamma_1\} \cup \{\gamma_2\} \cup \{\gamma_3\} \cup \{\gamma_4\}$ whose model based representation is $\Gamma^{(3)}$ and $\Gamma^{(4)}$, respectively. Clearly, these functions respond differently to the reasoning query $\alpha = (x_1 \vee \overline{x_2} \vee x_3 \vee x_4) \in 4$ -CNF, but this query cannot be detected by $\Gamma_{f|k}^B$. This shows that $\Gamma_{f|k}^B$ does not support correct reasoning with queries in 4-CNF, and implies that the additional knowledge that $\Gamma^{(3)}$ is in fact a set consisting of characteristic models does not help the reconstruction. ■

Theorem 4.2 provides a polynomial procedure only when k is a constant. The question arises therefore whether one can capture a larger set of conclusions, say $\log n$ -CNF, by projecting Γ in some other way. Clearly, in order to answer all the queries that are disjunctions of size k we need to have, or be able to generate, all projections of the elements in Γ_f^B on subsets of this size. We call these sets of variables, on which we project the elements of Γ_f^B , *windows* of variables. One such projection trivially exists, the projection over one window including the set of all variables. On the other hand, as we show, small windows are too restrictive.

We say that a window r_1 covers window r_2 if $r_2 \subseteq r_1$. Notice that in this case, we can answer queries on variables from the window r_2 using the projection over r_1 . Let R be a set of windows, and let L_k denote the set of $\binom{n}{k}$ windows of size k .

Claim 4.4 *If the largest window in R is smaller than \sqrt{n} then either $|R| \geq n^{k/2}$ or R does not cover all the windows in L_k .*

Proof: If the largest window in R is of size m , then every window in R covers at most $\binom{m}{k}$ windows of size k . Since $m < \sqrt{n}$, the number of windows we need in order to cover all the $\binom{n}{k}$ windows of size k is at least:

$$N = \frac{\binom{n}{k}}{\binom{m}{k}} > \sqrt{n}^k = n^{k/2}.$$

■

The last two claims suggest that model based reasoning with partial assignments is inherently limited. We can reason as long as the queries are in k -CNF for constant k but cannot go beyond that.

5 The Learning to Reason Framework

The Learning to Reason framework combines the interfaces to the world used by known learning models with the reasoning task and a performance criterion suitable for it. We first define the type of interactions available to the agent under the partial observation assumption, and then define the corresponding Learning to Reason tasks.

5.1 The Interface

We adapt standard definitions of oracles (Valiant, 1984; Angluin, 1988) to deal with partial assignments. As mentioned above we assume that one of the interpretations for partial assignments has been chosen, and is fixed for the duration of the learning scenario; when no confusion arises we omit the subscript denoting which interpretation has been chosen.

Definition 5.1 *An Example Oracle for a function f , with respect to the probability distribution D over $\{0, 1, *\}^n$, denoted $EX_D(f)$, is an oracle that when accessed, returns $(x, f(x))$, where x is drawn at random according to D .*

Definition 5.2 *A Membership Query Oracle for a function f , denoted $PMQ(f)$, is an oracle that when given an input $x \in \{0, 1, *\}^n$ returns $f(x)$. We denote the standard membership oracle, which answers only on total vectors, by $MQ(f)$.*

Definition 5.3 *An Equivalence Query Oracle for a function f , denoted $EQ(f)$, 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 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.*

Note that PMQ is a stronger oracle than MQ , since it answers all queries on total vectors and in addition all queries on partial vectors. On the other hand, EQ with partial assignments has more freedom than EQ with total assignments when choosing the counterexamples and is therefore weaker (since in addition to total assignments it can supply partial assignments). The next oracles, introduced in (Frazier and Pitt, 1993), can be thought of as using the reasoning process itself as a source for examples. An “entailment example oracle” similar to those was also used in (Greiner and Schuurmans, 1992).

Definition 5.4 An Entailment Membership Query Oracle for a function f , denoted $EnMQ(f, \mathcal{Q})$, is an oracle that when given as input a function $g \in \mathcal{Q}$ answers “Yes” if $f \models g$ and “No” otherwise.

Definition 5.5 An Entailment Equivalence Query Oracle for a function f , denoted $EnEQ(f, \mathcal{Q})$, is an oracle that when given as input a function g , answers “No” if and only if there exists a counterexample in \mathcal{Q} , namely, a function $h \in \mathcal{Q}$ such that, either $f \not\models h$ but $g \models h$ (a negative counterexample), or $f \models h$ but $g \not\models h$ (a positive counterexample). When it answers “No” it also supplies the counterexample.

We note that $EnEQ(f, \mathcal{Q})$ actually checks the equivalence of f and g relative to \mathcal{Q} . In fact, $EnEQ(f, \mathcal{Q})$ checks the equivalence of the least upper bounds of g and f in \mathcal{Q} . See (Selman and Kautz, 1996; Khardon and Roth, 1994b) for definitions and discussion of least upper bounds. Familiarity with this concept is not needed in the rest of this paper.

Below we define more oracles that we need to distinguish from the ones defined above in the on-line learning scenario. We hence identify the ones above:

Definition 5.6 We denote by $I(f)$ the interface available to the learner when learning to reason with respect to f ; this can be any subset of the oracles defined above.

5.2 The Learning to Reason Task

As in the known learning models we distinguish between Learning to Reason in a “batch” type scenario, and “on-line” Learning to Reason. To simplify notation, we assume from now on that all the functions discussed can be represented, in the corresponding representation class, with size polynomial in n (the number of variables), for some fixed polynomial.

Definition 5.7 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 $p(\cdot)$ 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 $f \models \alpha$.

When a probability distribution governs the occurrence of instances in the world we somewhat relax the requirements using the following restriction:

Definition 5.8 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 if $W \not\models \alpha$ but the weight of W outside α is very small, we may allow the algorithm to err (and answer that $W \models \alpha$). When W and D are clear from the context we will say that α is ϵ -fair.

Definition 5.9 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 $p(\cdot, \cdot)$ such that for any probability distribution D over $\{0, 1, *\}^n$, 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 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 presented to the algorithm as if presented by an oracle. Thus, a reasoning error may supply the algorithm with 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. The following oracle is used to model the learning interface for the on-line scenario.

Definition 5.10 *A Reasoning Query Oracle for a function f and a propositional 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 its 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”. We call the oracle a Reasoning Query Oracle with Counterexamples, denoted $RQC(f, \mathcal{Q})$, if when $f \not\models \alpha$ and a reasoning mistake is made, it supplies a counterexample (i.e., $x \in f \setminus \alpha$ where $x \in \{0, 1, *\}^n$).*

When learning, the algorithm is charged one mistake each time the reasoning query is answered incorrectly, and a successful learner should make a small number of mistakes.

Definition 5.11 *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.*

6 L2R with Partial Assignments

In this section we present the Learning to Reason results of the paper. In particular, we show that the oracle EX_D can be used to reason with respect to ϵ -fair queries, that the oracle RQ restricted to k -CNF queries can be used to reason with respect to the same class of queries, and that the stronger oracles, PMQ and entailment oracles, can be used to support reasoning with respect to all common queries.

6.1 A Sampling Approach

We first show that a simple sampling approach to reasoning, analyzed for total models in (Khardon and Roth, 1994a), works for partial assignments as well. The main difference is that, for partial assignments, the problem of model evaluation may be computationally hard (Claim 3.2). Therefore we have to restrict attention to queries that can be evaluated in polynomial time.

Recall that the distribution D used in EX_D is defined over $\{0, 1, *\}^n$. Let \mathcal{Q}_E be a class of functions that can be evaluated in polynomial time on partial assignments. Notice that this class depends on the interpretation of the partial assignments. For the existential interpretation, this class includes all DNF formulas, Horn CNF formulas, 2-CNF formulas, and $\log n$ -clause CNF formulas, but not general CNF representations.

The algorithm *Sample and Reason* first takes a sample of size $m = \frac{1}{\epsilon}(\ln |\mathcal{Q}_E| + \ln \frac{1}{\delta})$ examples from $EX_D(W)$. Let Γ be the set of positive examples sampled. Then, whenever presented with a

query, the algorithm uses Γ and the algorithm **MBR** to answer the query, where model evaluation is done according to the appropriate interpretation.

A simple probabilistic argument yields the learning result. Let $\alpha \in \mathcal{Q}_E$ be an ϵ -fair query, and assume that $W \not\models \alpha$, then the probability that no example x in Γ is such that $\alpha(x) = 0$ is at most $(1 - \epsilon)^m \leq e^{-(\ln |\mathcal{Q}_E| + \ln \frac{1}{\delta})} \leq \frac{\delta}{|\mathcal{Q}_E|}$. The probability that such an event happens for any query in \mathcal{Q}_E is therefore at most δ . Observing that if $W \models \alpha$ a model based reasoning algorithm cannot make a mistake on α (since it cannot find a counterexample which does not exist) we get the following theorem:

Theorem 6.1 *The algorithm Sample and Reason is a PAC-L2R algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q}_E)$ for any class \mathcal{F} .*

The sampling result presented above may be used to motivate an approach that views the interaction with the agent's environment as an integral part of the reasoning process; with few assumptions and using a simple interface between the agent and its environment, the L2R algorithm can support efficient reasoning under the restriction that queries are ϵ -fair and can be evaluated efficiently. These restrictions can be expressed and argued about since we combine learning and reasoning. The results presented next do not assume ϵ -fairness of queries.

6.2 Relations Among Oracles

Before presenting more learning results we discuss the relations among several oracles that have been defined.

Lemma 6.2 *The oracle $PMQ(W)$ for the existential interpretation can be simulated efficiently by the oracle $EnMQ(W, disjunctions)$, and vice versa.*

Proof: The lemma follows from the relations pointed out in Claim 3.2. Given a partial assignment y presented to PMQ we present the clause c_y (Definition 3.1) to $EnMQ(W, disjunctions)$ and reverse the (yes/no) answer received. The correctness follows from part (3) of Claim 3.2, namely, $W \models c_y$ iff $W_\epsilon(y) = 0$. On the other hand, given a query c presented to $EnMQ(W, disjunctions)$, we present the partial assignment y_c to PMQ and reverse the answer. Correctness follows from part (4) of Claim 3.2, namely, that $W \models c$ iff $W_\epsilon(y_c) = 0$. Clearly, in both cases the simulation is efficient. ■

It is also easy to see that the oracles $RQ(W, Q)$, and $EnEQ(W, Q)$ are closely related. Namely RQ is, in some sense, an on-line version of $EnEQ$; if the learning algorithm has a hypothesis h that it presents to $EnEQ$, then the counterexamples returned by $EnEQ$ are exactly those queries on which the algorithm will err, if presented by RQ . Thus, the information given to the learning algorithm is the same in both cases. A more subtle relation exists between $EnMQ$ and RQ in cases where the queries used by the RQ oracle are only those the algorithm is interested in asking itself. In this case one can use RQ instead of $EnMQ$. We make use of all these relations in the following results.

6.3 L2R with k -CNF queries

While the class of k -CNF queries is expressive, the implication relation for this class can be captured by simple enumeration. Namely, it is sufficient to know the implication relation for disjunctions of

size $\leq k$ and these can be easily combined to find the correct answer for any k -CNF expression. This follows from the fact that (for any W, c_1, c_2) if $W \not\models c_1 \wedge c_2$ then either $W \not\models c_1$ or $W \not\models c_2$. Therefore, given a CNF expression one can check the implication for every disjunction in it and answer “No” if one of these disjunctions is not implied by W .

Since the number of disjunctions of size $\leq k$ is bounded by $3^k \binom{n}{k}$ and is therefore polynomial for constant k , this leads to a polynomial algorithm. These observations have been used before (Moses and Tennenholtz, 1996) where off-line compilation of knowledge bases into this form is suggested. In contrast, we do not suggest compiling a knowledge base, a process that is computationally expensive, but instead learning the resulting representation directly.

Using this observation we can present two L2R algorithms, that given access to $EnMQ(W, k\text{-disjunctions})$ solve the reasoning problem $(\mathcal{F}, k\text{-CNF})$, for any class \mathcal{F} . The first algorithm, *A-IMP* runs through the list of all k -disjunctions and uses $EnMQ(W, k\text{-disjunctions})$ to collect a list of all k -disjunctions c such that $W \models c$. Then, given a query $\alpha = c_1 \wedge c_2 \dots \wedge c_m$, the algorithm says “Yes” ($W \models \alpha$) if and only if *all* the disjunctions c_1, c_2, \dots, c_m in α are in this list. The second algorithm, *A-NIMP* also runs through the list of all k -disjunctions, but uses $EnMQ(W, k\text{-disjunctions})$ to collect a list of all k -disjunctions c such that $W \not\models c$. Then, given a query $\alpha = c_1 \wedge c_2 \dots \wedge c_m$, the algorithm says “No” ($W \not\models \alpha$) if and only if at least one of the disjunctions c_1, c_2, \dots, c_m in α is in its list. We have:

Claim 6.3 *Both A-IMP and A-NIMP are E-L2R algorithms for the reasoning problem $(\mathcal{F}, k\text{-CNF})$ for any class \mathcal{F} . The number of queries the algorithms make is bounded by $3^k \binom{n}{k}$.*

For any W let $N(W)$ be the number of disjunction of size $\leq k$ which are not implied by W , $N(W) = |\{d \mid d \text{ has at most } k \text{ literals and } W \not\models d\}|$ and $P(W)$ the size of the complement set: $P(W) = |\{d \mid d \text{ has at most } k \text{ literals and } W \models d\}|$. Clearly, both $N(W)$ and $P(W)$ are smaller than the number of k -disjunctions.

The two algorithms presented above can be converted to on-line L2R algorithms, yielding a more natural view of the reasoning process as well as better bounds on the number of queries required. An on-line version for *A-IMP* proceeds as follows. Start with the empty list and initially predict “No”. If a mistake is made (that is, $W \models \alpha$) add all the clauses in α to the list. Reasoning is done as above; say “Yes” iff all the clauses in α are in the list. Clearly, this algorithm never makes mistakes when $W \not\models \alpha$, and the number of mistakes it makes is bounded by $P(W)$.

Claim 6.4 *There exists a MB-L2R algorithm for the reasoning problem $(\mathcal{F}, k\text{-CNF})$. The algorithm uses $RQ(W, k\text{-CNF})$ and the number of mistakes it makes is bounded by $P(W)$.*

A similar transformation for *A-NIMP* does not quite work since when a mistake is made such that $W \not\models \alpha$ all we know is that at least one of the clauses in α is not implied by W . To tackle that, all the clauses in α are added (temporarily) to the list, to be removed later if they contribute to mistakes of the other kind, that is, when $W \models \alpha$. In fact, instead of using a list of clauses our algorithm maintains a list of partial assignments and uses the algorithm **Lazy-MBR** to answer queries. This reduces the number of mistakes further since if $c' \models c$, and we have a counterexample y such that $c(y) = 0$ that can be detected by **Lazy-MBR**, then it is also true that $c'(y) = 0$ and hence we can conclude that $W \not\models c'$.

The algorithm *A-OL-NIMP*, described in Figure 4, maintains a model based representation G , initially empty, and uses the lazy evaluation algorithm **Lazy-MBR** (Figure 2) to respond to queries presented by RQ . Let $\alpha = c_1 \wedge c_2 \dots \wedge c_m$ be the CNF representation of a query supplied by RQ , and assume that the algorithm makes a mistake on α . When $W \not\models \alpha$ and the algorithm

Algorithm A-OL-NIMP

1. Initialize $G = \emptyset$.
2. $\alpha \leftarrow RQ(W, \mathcal{Q})$
3. Use **Lazy-MBR**(G, α) to answer the query $W \models \alpha$.
4. If “wrong” then:
 - if Lazy-MBR answered No** then for all $z \in G$ and for all $c \in \alpha$ if z falsifies c then remove z from G .
 - if Lazy-MBR answered Yes** then for all $c \in \alpha$, compute the assignment $y_c \in \{0, 1, *\}^n$ (as in Definition 3.1), and if it was never added to G before then add y_c to G .
5. GoTo 2

Figure 4: The Algorithm A-OL-NIMP

responded “Yes”, the algorithm produces a set of assignments and adds them to G . For each $c_i \in \alpha$, the algorithm produces the assignment y_{c_i} as in Definition 3.1. (For example, if $c = x_1 \vee \overline{x_3}$ then $y_c = (0 * 1)$.) The assignment is added to G only if it was never considered before. When $W \models \alpha$ and the algorithm responded “No”, the algorithm removes from G the assignments that caused the mistake.

It is important to notice that, while the algorithm uses partial assignments, the following theorem is independent of the interpretation of the partial assignments. The reason is that it does not receive any examples from the oracles; those are produced internally by the algorithm.

Theorem 6.5 *For any class \mathcal{F} of Boolean functions, the algorithm A-OL-NIMP, when given access to $RQ(W, k\text{-CNF})$, is a MB-L2R algorithm for the reasoning problem $(\mathcal{F}, k\text{-CNF})$. The number of mistakes A-OL-NIMP makes is bounded by $\min\{m \cdot N(W), 3^k \binom{n}{k}\}$, where m is the maximal number of clauses in a query presented to the algorithm.*

Proof: It is sufficient to keep in G a (useful) counterexample for every disjunction c with up to k variables such that $W \not\models c$.

For every mistake the algorithm makes when $W \not\models \alpha$, there must be a $c \in \alpha$ such that $W \not\models c$. Whenever a mistake like this happens the algorithm adds an assignment y_c for every disjunction $c \in \alpha$. At least one of these assignments is a “good counterexample”, in the sense that it falsifies the disjunction $c \in \alpha$ such that $W \not\models c$, and is thus useful for the **Lazy-MBR** algorithm. (Notice that by “falsifying a disjunction” we mean falsifying all the variables in the disjunction; this is obviously true by the construction.)

The way the assignments are generated guarantees that the algorithm will not make a mistake on any query which contains this clause, and since the algorithm uses **Lazy-MBR** for reasoning, correct counterexamples are never removed from G . We therefore get a bound of $N(W)$ for the number of mistakes in which the algorithm says “Yes” instead of “No”. Every such mistake is responsible for producing at most $(m-1)$ mistakes of the other type. Since the algorithm introduces every clause at most once, we get the claimed mistake bound. ■

Corollary 6.6 *For any class \mathcal{F} of Boolean functions, there is a MB-L2R algorithm for the reasoning problem $(\mathcal{F}, k\text{-CNF})$ that uses $RQ(W, k\text{-CNF})$, and makes at most $2 * \min\{m \cdot N(W), P(W)\}$ mistakes, where m is the maximal number of clauses in a query presented to the algorithm.*

Proof: The algorithm simply alternates between the algorithm used in Theorem 6.5 and the algorithm used in Claim 6.4. When the currently used algorithm makes a mistake we update its representation and go on to use the other algorithm. This clearly results in the stated mistake bound. ■

We note that for the universal and the abbreviated interpretations, it is possible to avoid the dependence on m in the above bound, by using RQC instead of RQ . For these interpretations the counterexamples supplied by the oracle are useful⁴ for model based reasoning. The dependence of the mistake bound can be reduced in a different way, which holds for all the interpretations. This is done by converting the algorithm $A\text{-OL-NIMP}$ to an attribute-efficient algorithm using the Winnow algorithm (Littlestone, 1988), similar to what is done in (Blum, 1992). The resulting mistake bound is of the form $\log m \cdot N(W)$. The knowledge representation used by the new algorithm is not a list of models any more. Instead, a weighted sum of models is used for reasoning, where the weights are a function of the learning history.

The results of this section demonstrate the utility of the framework in that one can learn to reason even when the learning problem and the reasoning problem are hard. In particular, if W is an arbitrary Boolean circuit, the learning problem is known to be hard (independent of the representation (Kearns and Valiant, 1994)), and the reasoning problem is also hard, regardless of the class of queries (Cook, 1971). However, by restricting the queries to $k\text{-CNF}$ we show that one can learn to reason. Finally, we note that in ILP terminology (De Raedt, 1997) the above shows that $k\text{-CNF}$ is learnable from entailment.

6.4 Using Stronger Oracles

The results of Section 4.2 suggest that the only way to improve the results we have shown so far, and reason with classes of queries that are wider than $k\text{-CNF}$, is to use total models in the representation. On the other hand, since the interface may supply short examples, and reconstruction is not possible, this seems to be impossible. In order to learn to reason with more expressive queries, we use stronger oracles, that can be used to collect the set of (total) characteristic models Γ . We use a Learning to Reason result that was established in the context of total models. We show that the same task can be performed using the appropriate partial assignments oracles. Recall that by \mathcal{Q}_C we denote the class of all common queries.

Theorem 6.7 (Khardon and Roth, 1994a) *There is a MB-L2R algorithm, $Ex\text{-L2R-DNF}$, for the reasoning problem $(\mathcal{F}, \mathcal{Q}_C)$, The algorithm interacts with the oracles $RQC(W, \mathcal{Q}_C)$ and $MQ(W)$ restricted to total models, maintains a model based representation $G \subseteq W$ (where W is the hidden world function), and when presented a query by RQC , it responds using G and the model based algorithm **MBR**. The algorithm is polynomial in the DNF size of W and the size of the queries presented to it.*

Theorem 6.8 *There is a MB-L2R algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q}_C)$, for any class \mathcal{F} and for all interpretations of partial assignments.*

⁴This does not hold for the existential interpretation since the counterexample can be adversarially chosen, such that **Lazy-MBR** will make repeated mistakes on the same query.

(1) For the abbreviated and universal interpretations, the algorithm uses the oracles $RQC(W, \mathcal{Q}_C)$ and $MQ(W)$.

(2) For the existential interpretation the algorithm uses the oracles $RQ(W, \mathcal{Q}_C)$ and $PMQ(W)$.

The algorithm is polynomial in the DNF size of W and the size of the queries presented to it.

Proof: We show that the partial assignments oracles available here are as powerful as the total assignments oracles used in Theorem 6.7. We can therefore simulate algorithm *Ex-L2R-DNF* and hence learn to reason.

Clearly, a membership query oracle for partial assignments can answer all membership queries on total assignments. Instead of the oracle RQC for total assignments as in *Ex-L2R-DNF* we have access to either RQC with partial assignments in case (1) or to RQ in case (2). Therefore, we can use the given oracle for the purpose of interacting with the algorithm; what we need to do is to find a *total* counterexample to present to *Ex-L2R-DNF* when it makes a mistake.

Let α be a query on which the algorithm makes a mistake when interacting with RQ . First note that since the algorithm performs model based reasoning, and the models in its representation are true models of W , it must be the case that the algorithm said “Yes” whereas in fact $W \not\models \alpha$.

We argue by cases, according to the interpretation used. For the universal and abbreviated interpretations, we have access to RQC . That is, whenever a mistake is made, RQC returns a partial assignment $v \in W \setminus \alpha$ as a counterexample. In the abbreviated case, the unique 0-padded extension is the total counterexample needed. In the universal case, by definition, v must have an extension which falsifies a disjunction $d \in \alpha$. Such an extension is easy to find by finding a clause in α in which non of the literals is satisfied by v . (By definition, this extension satisfies W .)

For the existential interpretation, we have access to RQ , and therefore need to generate a total counterexample on our own. As in Theorem 6.5, we generate the counterexamples using the structure of the query α . Given $\alpha = \wedge c_i$, we generate the partial examples y_{c_i} . At least one of these examples, $v = y_{c_j}$, for some j , is positive for W . That is, the partial vector v is a counterexample. Next we show how, using PMQ , and these partial examples we generate a total counterexample. First, using PMQ we find $v = y_{c_j}$ which is positive for W . Then, we extend v to a total vector which is still a counterexample. This can be done in a greedy manner, one bit at a time, using the oracle PMQ . Let v be the counterexample from above. Then, by definition, there exists some total example which is an extension of v and is positive for W . If there is a positive extension with 0 assigned to some bit, which is currently assigned *, (use a PMQ query to test this) then we can assign 0 to it, otherwise we can assign 1 to it. ■

It is important to notice that the oracle PMQ is rather strong in that in the proof above it enabled the completion of partial assignments to total assignments.

6.5 Using Entailment Queries

Entailment oracles were introduced in Frazier and Pitt (1993) where an algorithm for learning to classify Horn expressions is developed. As argued in (Khaddon and Roth, 1994a) this algorithm can be used as a learning to reason algorithm for the class of Horn queries and arbitrary W (and is polynomial for a certain class of functions discussed below). We show that entailment oracles can be used to learn the set of characteristic models, and therefore learn to reason with a larger set of queries. However, as explained below, the two results are incomparable, since both the complexity of the algorithms and the strength of the oracles used are incomparable.

Theorem 6.9 *There is an E-L2R algorithm for the reasoning problem $(\mathcal{F}, \mathcal{Q}_C)$, for any class \mathcal{F} , when given access to $EnEQ(W, \mathcal{Q}_C)$ and $EnMQ(W, disjunctions)$. The algorithm is polynomial in n , the size of the DNF representation of W and the size of the queries presented to it.*

Proof: As in Theorem 6.8 the proof relies on simulating the oracles MQ and RQC used in Theorem 6.7. First note that Lemma 6.2 implies that using $EnMQ(W, disjunctions)$ one can simulate $PMQ(W)$, in the case of the existential interpretation. In particular, one can simulate MQ using $EnMQ$.

We now show that $EnEQ$ can replace the oracle RQC . Recall that the algorithm uses a model based representation with respect to a basis B . We will use the function $h = \bigwedge_{b \in B} \bigvee_{z \in G} \mathcal{M}_b(z)$ as the hypothesis of the algorithm. When the algorithm tries to access RQC , we instead call $EnEQ(W, \mathcal{Q}_C)$ with the hypothesis h , and receive a counterexample $\alpha \in \mathcal{Q}_C$. We claim that only one type of counterexample can occur. Namely, it must be the case that $h \models \alpha$, and $W \not\models \alpha$. To prove this claim⁵, assume that $h \not\models \alpha$. Then, by Theorem 4.1 there is a minimal assignment x of h such that $\alpha(x) = 0$. Furthermore, x must be an element of G , since for all $b \in B$, $\exists z \in G$ such that $x \geq_b z$. But then since $G \subseteq W$ we also get $W \not\models \alpha$.

We present α that was received from $EnEQ$ to the algorithm, as if coming from RQC . Since the algorithm performs model based reasoning with G , by Theorem 4.1, it will make a mistake on α . We then compute a counterexample and return it to the algorithm, along with the response “No”. Computing the counterexample is done exactly in the same manner as in the proof of Theorem 6.8, where PMQ for the existential interpretation is used.

Finally, note that when the algorithm stops it responds correctly to all queries $\alpha \in \mathcal{Q}_C$. By the definition of $EnEQ$, when the algorithm stops (i.e., when $EnEQ$ returns “Yes”), $W \models \alpha$ iff $h \models \alpha$, for all $\alpha \in \mathcal{Q}_C$. Therefore, by Theorem 4.1, model based reasoning with G is correct. ■

We note that the strength of the oracles used is incomparable with those used by Frazier and Pitt (1993). When using $EnMQ$, we used a “stronger” version of it, since we allowed the algorithm to ask queries that are arbitrary disjunctions, compared to Horn disjunctions used there. On the other hand we used a “weaker” version of $EnEQ$ since the oracle was allowed to return counterexamples in \mathcal{Q}_C , compared with Horn disjunctions used there. The results are also incomparable in complexity since the algorithm in Frazier and Pitt (1993) requires time which is polynomial in the propositional Horn representation of the least upper bound of W , whereas Theorem 6.9 requires time polynomial in the DNF size of W , and the two are in general incomparable. Another important difference relates to partial assignments. As discussed in the previous section the oracles used here, PMQ and also $EnEQ$ can give more information than their total-assignment counterparts. This is different from the algorithm of Frazier and Pitt (1993) which always asks $EnMQ$ queries which are shorter than the disjunctions it receives as counterexamples.

7 Conclusions

The Learning to Reason framework combines the study of Learning and Reasoning into a single task. Within this framework, learning is done specifically for the purpose of reasoning with the learned knowledge. Computational considerations show that this is a useful paradigm; in some cases

⁵This can be traced to the fact that $h \models W_{lub}^B$, where W_{lub}^B is the least upper bound of W in \mathcal{Q}_C (Khardon and Roth, 1994b). We give an argument from first principles to avoid using the notion of least upper bounds.

learning and reasoning problems that are intractable when studied separately become tractable when performed as a task of Learning to Reason.

In this paper we considered the problem of Learning to Reason when the interaction with the world supplies only partial information in the form of partial assignments. Several natural interpretations of partial assignments were considered, and techniques for reasoning with models were extended to deal with partial assignments. The Learning to Reason tasks were studied with respect to these interpretations. We studied several interfaces of the agent with its environment that are suitable to handle partial assignments, and have proved Learning to Reason results whose strength depend on the type of interaction assumed. In particular, random partial assignments can be used to reason with respect to ϵ -fair queries, the oracle RQ restricted to k -CNF queries can be used to reason with respect to the same class of queries, and stronger oracles can be used to support reasoning with respect to all common queries.

The computational advantages of L2R were explored and discussed. For example, for the case of common queries, similar to (Khardon and Roth, 1994a), the results in this paper show that while learning DNF is still an open problem, one can learn to reason for domains with a polynomial size DNF.

Finally, our model is also related to ILP, where some of the work can be viewed (in our terminology) as “learning to reason with a specific knowledge representation”, and some of our results can be understood (in ILP terminology) as performing learning from entailment. One major difference between these frameworks is that work in ILP is mainly concerned with first order logic representations while we have discussed propositional logic. Further exploration of these relations and their possible implications is an interesting direction for future work; some preliminary ideas are reported in (Khardon, 1998).

Acknowledgments

We are grateful to Moti Frances, Wolfgang Maass, Les Valiant and anonymous referees for their useful comments on earlier versions of this paper.

References

- Angluin, D. 1988. Queries and concept learning. *Machine Learning*, 2(4):319–342, April.
- Ben-David, S. and E. Dichterman. 1993. Learning with restricted focus of attention. In *Proc. of the Annual ACM Workshop on Computational Learning Theory*, pages 287–296. ACM Press, New York, NY.
- Blum, A. 1992. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, October.
- Blum, A., L. Hellerstein, and N. Littlestone. 1991. Learning in the presence of finitely or infinitely many irrelevant attribute. In *Proc. of the Annual ACM Workshop on Computational Learning Theory*, pages 155–166.
- Bshouty, N. H. 1995. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153.
- Cook, S. A. 1971. The complexity of theorem proving procedures. In *3rd annual ACM Symposium of the Theory of Computing*, pages 151–158.

- De Raedt, L. 1997. Logical settings for concept learning. *Artificial Intelligence*, 95(1):187–201.
- Frazier, M. and L. Pitt. 1993. Learning from entailment: An application to propositional Horn sentences. In *Proc. of the International Conference on Machine Learning*, pages 120–127. Morgan Kaufmann.
- Greiner, R., A. Grove, and A. Kogan. 1996. Exploiting the omission of irrelevant data. In *Proc. of the International Conference on Machine Learning*, pages 216–224.
- Greiner, R., A. Grove, and D. Roth. 1996. Learning active classifiers. In *Proc. of the International Conference on Machine Learning*, pages 207–215.
- Greiner, R. and D. Schuurmans. 1992. Learning useful Horn approximations. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 383–392.
- Kautz, H., M. Kearns, and B. Selman. 1995. Horn approximations of empirical data. *Artificial Intelligence*, 74:129–145.
- Kearns, M.J. and L.G. Valiant. 1994. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95.
- Khardon, R. 1996. Learning to take actions. In *Proc. of the National Conference on Artificial Intelligence*, pages 787–792.
- Khardon, R. 1998. Learning first order universal Horn expressions. In *Proc. of the Annual ACM Workshop on Computational Learning Theory*. Forthcoming.
- Khardon, R. and D. Roth. 1994a. Learning to reason. In *Proc. of the National Conference on Artificial Intelligence*, pages 682–687. To appear in *Journal of the ACM*.
- Khardon, R. and D. Roth. 1994b. Reasoning with models. In *Proc. of the National Conference on Artificial Intelligence*, pages 1148–1153.
- Khardon, R. and D. Roth. 1997. Default and relevance in model based reasoning. *Artificial Intelligence*, 97(1-2):169–193.
- Littlestone, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318.
- Moses, Y. and M. Tennenholtz. 1996. Off-line reasoning for on-line efficiency: knowledge bases. *Artificial Intelligence*, 83(2):229–239.
- Muggleton, S. and L. De Raedt. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 20:629–679.
- Naor, J. and M. Naor. 1993. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, August.
- Reiter, R. 1987. Nonmonotonic reasoning. In *Annual Reviews of Computer Science*. Annual Reviews Inc., pages 147–188.
- Roth, D. 1995. Learning to reason: The non-monotonic case. In *Proc. of the International Joint Conference of Artificial Intelligence*, pages 1178–1184.

- Schuermans, D. and R. Greiner. 1994. Learning default concepts. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence (CSCSI-94)*, pages 519–523.
- Selman, B. and H. Kautz. 1996. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, March.
- Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November.
- Valiant, L. G. 1995. Rationality. In *Workshop on Computational Learning Theory*, pages 3–14, July.