PAC + SMT

Ionela G. Mocanu
University of Edinburgh
i.g.mocanu@ed.ac.uk

Vaishak Belle
University of Edinburgh &
Alan Turing Institute
vaishak@ed.ac.uk

Brendan Juba
Washington University in St. Louis
bjuba@wustl.edu

Abstract

To deploy knowledge-based systems in the real world, the challenge of knowledge acquisition must be addressed. Knowledge engineering by hand is a daunting task, so machine learning has been widely proposed as an alternative. However, machine learning has difficulty acquiring rules that feature the kind of exceptions that are prevalent in real-world knowledge. Moreover, it is conjectured to be impossible to reliably learn representations featuring a desirable level of expressiveness. Works by *Khardon and Roth* and by *Juba* proposed solutions to such problems by learning to reason directly, bypassing the intractable step of producing an explicit representation of the learned knowledge. These works focused on Boolean, propositional logics. In this work, we consider such implicit learning to reason for arithmetic theories, including logics considered with satisfiability modulo theory (SMT) solvers. We show that for standard fragments of linear arithmetic, we can learn to reason efficiently. These results are consequences of a more general finding: we show that there is an efficient reduction from the learning to reason problem for a logic to any sound and complete solver for that logic.

1 Introduction

To deploy knowledge-based systems in the real world, the challenge of knowledge acquisition must be addressed. Knowledge engineering by hand is a daunting task, so machine learning has been widely proposed as an alternative. In that regard, standard techniques such as inductive logic programming (ILP) [18] and concept learning [22] have been very influential in the area. Moreover, in a logical context, Valiant [21] recognized that the challenge of learning should be integrated with deduction. In particular, he proposed a semantics to capture the quality possessed by the output of (probably approximately correct) PAC-learning algorithms when formulated in a logic. Although weaker than classical entailment, it allows for a powerful model-theoretic framework for answering queries.

What concerns us in this work is the extension of these frameworks to fragments of first-order logic commonly used for representing continuous domains, such as the logics considered with *satisfiability modulo theory* (SMT) solvers [1]. SMT has been widely used for applications such as model checkers, verification, unit test generators, interactive theorem provers for higher-order logic, as well as probabilistic inference [3]. Specifying the appropriate domain constraints can be very challenging, owing to the numeric nature of the underlying language. Somewhat surprisingly, there is very little work that addresses this gap. To the best of our knowledge, Kolb et al. [15] were the first to tackle this issue and propose an approach to find a set of constraints that is consistent with a given data set. However, this may be an over-approximation, and there is no guarantee for how close these constraints are to characterizing the actual set of solutions. Moreover, this approach requires a noiseless data set. So we ask the question: *can we provide any guarantees for robust learning of SMT formulas?*

From the standpoint of learning an expressive logical knowledge base and reasoning with it, most PAC results are somewhat discouraging. In particular, machine learning has difficulty acquiring rules that feature the kind of exceptions that are prevalent in real-world knowledge. For example, in agnostic learning [13] where one does not require examples (drawn from an arbitrary distribution) to be fully consistent with learned sentences, efficient algorithms for learning conjunctions would yield an efficient algorithm for PAC-learning disjunctive normal form (DNF) formulas (also over arbitrary distributions), which current evidence suggests to be intractable [9]. Works by Khardon and Roth [14] and by Juba [12] proposed solutions to such problems by learning to reason directly, bypassing the intractable step of producing an explicit representation of the learned knowledge. Thus, there is no "discovery" of the representation, which also means that no syntactic biases are necessary, beyond the assumption that the hypothesis is drawn from the same language as the examples and the background knowledge. It is perhaps also interesting to contrast this line of work with (standard) ILP: the latter searches for a hypothesis H (a set of formulas) that is consistent with the examples by appealing to entailment. (See table 1.) It does not, however, seek to analyze the degree to which the resulting formulas capture an unknown, ground-truth process that produced the examples.

Learning model	**PAC Learnability**	**ILP**
Language	Propositional [12], FOL+ equality [4], SMT (current)	Propositional, FOL
Examples	Partial interpretations $\{\rho^{(1)}, ..., \rho^{(m)}\}$	positive examples P (but also negative examples N)
Hypothesis learning paradigm	explicit [14], implicit ([12], [4], current)	explicit
Schema/Framework	Given KB, ρ and α, is it true that $KB \cup \text{implicit } H \models \alpha$	Given KB and P find H s.t. $KB \cup H \models P$

Table 1: Comparison of learning approaches

In this work, we show how to extend the implicit learning approach to SMT formulas, yielding agnostic (implicit) learning of SMT formulas for the purposes of deciding entailment queries. Specifically, we first describe extensions of the definitions and operations underlying the implicit learning technique to SMT formulas and continuous-valued data. Second, we establish in general that implicit learning can be done for fragments of logical languages that are closed under substitutions of values for variables and possess sound and complete decision procedures. We do this by showing that completeness plus the closure under substitutions of the language implies the key "restriction-closure" property underpinning Juba's framework [12]. This turns out to be convenient: we finally note that since many popular fragments of SMT are known to have such sound and complete decision procedures, we immediately obtain such constraint learning for these fragments.

In summary, our results contribute to a theoretical understanding of the PAC learnability of logical theories. The PAC learning of Boolean functions (i.e., formulas), which corresponds to the "explicit" discovery of a hypothesis, is only feasible for rather simple representations. As we say, learning of CNF/DNF itself is likely intractable. Implicit learning, in contrast, simply focuses on answering queries without trying to explicitly identify the hypothesis (set of KB rules), which is the computational bottleneck. The algorithm gets as input the background knowledge and a finite set of partial assignments. The goal is to then decide entailment of the query using both the KB and the partial assignments. In other words, in the explicit approach, a formula is learned from the assignments, and entailment judgements are then computed from that formula, but in the implicit approach, the partial assignments are used to determine where queries are entailed.

2 Formal Framework

2.1 Logical background

Satisfiability (SAT) is the problem of deciding whether there exists an assignment of truth values (i.e., model) to variables (propositional symbols) such that a propositional logical formula α is true. *Satisfiability modulo theories* (SMT) is a generalization to SAT for deciding satisfiability for fragments and extensions of first-order logics with equality, for which specialized decision procedures have been developed. Deciding satisfiability for these logics is done with respect to some decidable background theory which fixes the interpretations of functions and predicates [1]. In this work, we are especially interested in the background theories of quantifier-free arithmetic over the integers and over the reals. The following formal exposition of the logical language is adapted from [1].

Syntax: We assume a logical signature consisting of the set of predicates denoted as $\mathcal{P}$, and a set of functions symbols $\mathcal{F}$, including 0-ary functions, logical variables, and standard connectives. An atomic formula is one of the form: a (a propositional symbol), $pred(t_1, ..., t_k)$, $t_1 = t_2$, $\perp$ (false), $\top$ (true). A literal l is an atomic formula or its negation $\neg l$. A clause is a disjunction $l_1 \vee ... \vee l_k$ of literals. We denote clauses as c (with superscripts and subscripts) and identify the empty clause with the formula $\perp$. A ground expression is one where all the variables are replaced by the domain of discourse (e.g., integers, reals, finite set of named objects).

Semantics: In terms of meaning, formulas are given a truth value from the set $\{true, false\}$ by means of first order models. A model $\boldsymbol{\rho}$ is a pair consisting of a non-empty set Σ, the universe of the model and a mapping assigning to each constant symbol a an element $a \in \Sigma$ (the domain), to each function symbol $f \in \mathcal{F}$ of arity $n > 0$ a total function $f : \Sigma^n \to \Sigma$, to each propositional symbol b an element $b \in \{true, false\}$ and to each predicate $p \in \mathcal{P}$ of arity $n > 0$ a total function $p : \Sigma^n \to \{true, false\}$. (A partial model will be written using the regular font as ρ vs the bold font for a full model as $\boldsymbol{\rho}$.) Terms are interpreted as usual, as is the satisfaction relation that is defined inductively. As discussed above, we assume satisfaction and entailment wrt a suitable background theory (e.g., theory of reals with the understanding that inequalities and other mathematical operators are interpreted as usual). See [1] for details.

2.2 PAC semantics and the learning model

Inductive generalization (as opposed to deduction) inherently has to cope with mistakes. Thus, the kind of knowledge produced by learning algorithms cannot hope to be valid in the traditional (Tarskian) sense, except in extreme cases, such as assuming we see every data point in a noise-free manner. The PAC semantics was introduced by Valiant [21] to capture the quality possessed by the output of PAC-learning algorithms when formulated in a logic. Essentially, it is a relaxed semantics to capture the quality possessed by knowledge learned from independently drawn examples. The relaxed notion of *validity* that formulas may satisfy is then also defined in terms of the same distribution D used to produce the examples.

Definition 1: [$1 - \epsilon$-validity [12]] Given a distribution D over Σ^n, we say that a Boolean function b is $(1 - \epsilon)$-valid if $\Pr_{\boldsymbol{\rho} \in \Sigma^n}[b(\boldsymbol{\rho}) = 1] \geq 1 - \epsilon$. [1]

The notation and formulation introduced in [12] applies immediately to our setting, despite the fact that over the domain of reals, we are dealing with infinitely many models and so we will need a continuous distribution. We will clarify such subtleties as and when they are introduced. Note that, for simplicity, throughout the paper we assume that the language consists of n 0-ary function symbols, and so we are dealing with expressions of the form $x < y, x + y > 10, 2 \cdot x > z$, etc.

Previously, [12] would define a discrete distribution over $\{0, 1\}^n$, but we lift this to Σ^n, where Σ may be the set of reals $\mathbb{R}$, in which case any single model $\boldsymbol{\rho}$ would be assigned a density by Pr, and so we are saying the probability of the region satisfying b would be $\geq 1 - \epsilon$. For example, suppose that all points in the region $0 \leq x_1 \leq 4$ are accorded a density of 0.25, and b denotes the formula $x_1 \leq 2$, then $\Pr_{\boldsymbol{\rho} \in \mathbb{R}}[b(\boldsymbol{\rho}) = 1] = 0.5$, so we say $[x_1 \leq 2]$ is 0.5-valid.

We consider the reasoning problem of deciding whether or not a query formula α is $(1 - \epsilon)$-valid with respect to a data distribution D. We suppose we also have an explicit knowledge base Δ, where we generally presume Δ is 1-valid, i.e., satisfied on the support of D. If examples are drawn from D, Hoeffding's inequality guarantees that with high probability, the proportion of times that the query formula/input evaluates to *true* is a good estimate of the degree of validity of that formula.

[1]We remark that this work is in the context of PAC-semantics as opposed to the traditional PAC learning model which focuses on learning the classifier robustly. PAC semantics refers to $1 - \epsilon$ validity of formulas; the connection between the two is that if we use a PAC-learning algorithm to produce a rule $f(x)$ that predicts the value y, then the formula $f(x) = y$ is $(1 - \epsilon)$-valid (with probability $1 - \delta$).

Theorem 2: *[Hoeffding's inequality] Let $X_1, \dots, X_m$ be independent random variables taking values in* $[0, 1]$. *Then for any* $\epsilon > 0$*,*

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m} X_i \geq \mathbb{E}\left[\frac{1}{m}\sum_{i=1}^{m} X_i\right] + \epsilon\right] \leq e^{-2m\epsilon^2}.$$

For complete observations, that is when the algorithm is provided with full assignments of the variables used, we could therefore approximately decide $1 - \epsilon$-validity directly, distinguishing formulas that are $1 - \epsilon$-valid from those that are not $1 - \epsilon - \gamma$-valid for any desired $\gamma > 0$ given enough data. In practice, we are often interested in queries that refer to values or properties that are not explicitly represented in the data. In other words, the algorithm only gets to see partial models which is the case we focus on. So, instead of drawing our samples directly from the distribution D, the algorithm will receive information about D in the form of partial assignments drawn from a masking process introduced below.

Definition 3: [Masking process [17]] A *mask* is a function $M : \Sigma \rightarrow \Sigma \cup \{*\}$, with the property that for any $\boldsymbol{\rho} \in \Sigma^n$, $M(\boldsymbol{\rho})$ is *consistent* with $\boldsymbol{\rho}$, i.e., whenever $M(\boldsymbol{\rho})_i \neq *$ then $M(\boldsymbol{\rho})_i = \boldsymbol{\rho}_i$. We refer to elements of the set $(\Sigma \cup \{*\})^n$ as *partial assignments*. A *masking process* $\boldsymbol{M}$ is a mask-valued random variable. As in [12], we denote the distribution over partial examples obtained by applying the masking process as $\boldsymbol{M}(D)$.

In this way, a full model, describing the state of the world becomes an observation or a partial assignment by applying the masking process to it. Once we have the partial assignments we can attempt to evaluate a formula α on the partial assignment obtained from the masking process. If evaluation produces a Boolean value *true* or *false*, then we will say that this formula is *witnessed* in the partial assignment. Otherwise we will call the result of our partial evaluation a *restricted formula*:

Definition 4: [Restriction and witnessed formulas] Given a formula α and a partial assignment ρ, the restricted formula, denoted by $\alpha|_\rho$ is inductively defined as follows:

- If α is an atomic formula and none of the terms are given value $*$ in ρ, then $\alpha|_\rho$ is the formula representing the value that α evaluates to under the assignment given by ρ, and we say that α is *witnessed*. Otherwise, $\alpha|_\rho$ is given by substituting the assignments ρ_i for variables not given value $*$ by ρ.
- If $\psi = \neg\alpha$ and α is not witnessed in ρ, then $\alpha|_\rho = \neg(\psi|_\rho)$; otherwise, ψ is witnessed and takes the negation of the value of $\alpha|_\rho$.
- If $\psi = l_1 \vee \dots \vee l_n$ is a clause, if any $l_i|_\rho$ is witnessed true, $\psi|_\rho$ is also witnessed true; if every $l_i|_\rho$ is witnessed false, $\psi|_\rho$ is also witnessed false. And finally, otherwise $\psi|_\rho = (l_1|_\rho) \vee \dots \vee (l_k|_\rho)$
- For a restriction ρ and a set of formulas F, we let $F|_\rho$ denote the set $\{\alpha|_\rho : \alpha \in F\}$.

We have modified the definition slightly from [12]; there, the partial examples consisted of sets of values of atomic formulas directly, as opposed to values for the free variables, so our treatment of atomic formulas is different. Witnessed formulas correspond to the implicit knowledge base. We use partial models to simplify complex formulas in the Δ and query in order to capture the inferences in the knowledge base. As shown in [12], this is sufficient whenever the reasoning algorithm is "restriction closed:"

Definition 5: (Restriction closure) We say that a procedure A is restriction closed if and only if $\Delta \models \alpha$ implies that A proves $\alpha|_\rho$ from $\Delta|_\rho$, for any partial assignment ρ. Essentially, from any derivation of α from Δ by A, there is a proof of $\alpha|_\rho$ from $\Delta|_\rho$ by A.

Precisely then, we have the following:

Theorem 6: *[Implicit learning [12]] Let Δ be a conjunction of constraints representing the knowledge base and an input query α. We draw at random $m = \frac{1}{2\gamma^2} ln\frac{1}{\delta}$ partial assignments $\{\rho^{(1)}, \rho^{(2)}, ..., \rho^{(m)}\}$ from $\boldsymbol{M}(D)$ for the distribution D and a masking process $\boldsymbol{M}$. Suppose that we have a sound, restriction-closed decision procedure A. Then with probability* $1 - \delta$*,*

- *if* $(\Delta \Rightarrow \alpha)$ *is not* $(1 - \epsilon - \gamma)$ *- valid with respect to the distribution D, Algorithm 1 returns Reject; and*
- *if there exists some KB I such that* $\Delta \wedge I \models \alpha$ *and I is witnessed true with probability at least* $(1 - \epsilon + \gamma)$ *on* $\boldsymbol{M}(D)$*, then Algorithm 1 returns Accept.*

Moreover, if A runs in polynomial-time, so does Algorithm 1.

Algorithm 1: Implicit learning reduction

```
Input: Algorithm A, formula $\alpha$, variables $\epsilon, \delta, \gamma \in (0, 1)$, list of partial assignments
       $\{\rho^{(1)}, \rho^{(2)}, ..., \rho^{(m)}\}$ list of hypothesis formulas (Knowledge base) $\Delta$
Output: Accept if there exist proof of $\alpha$ in $S$ derivation proof from $\Delta$ and formulas $\phi_1, \phi_2$, ... that
        are simultaneously witnessed true with probability at least $(1 - \epsilon + \gamma)$-valid on $M(D)$
        Reject if $\Delta \Rightarrow \alpha$ is not $(1 - \epsilon - \gamma)$-valid under $D$
begin
    $B \leftarrow \lfloor \epsilon \times m \rfloor, FAILED \leftarrow 0$.
    foreach $i$ in $m$ do
        if $A(\alpha|_{\rho^{(i)}}, \Delta|_{\rho})$ returns UNSAT then
            Increment FAILED. if $FAILED > B$ then
                return Reject
    return Accept
```

Although we have modified the definitions of $(1 - \epsilon)$-validity and witnessing slightly, the proof remains the same. The S symbol represents the set of derivation steps, for example, given a constraint $(2x + 3) = 7$ a derivation proof S would look like: $S = \{2x = 7 - 3, 2x = 4, x = 4/2, x = 2\}$.

3 Implicit Learnability from Completeness

We now turn to considering implicit learning of linear arithmetic. In other words, the knowledge base Δ contains conjunctions of linear (or non-linear) inequalities. We also suppose we are given a set of partial assignments $\{\rho^{(1)}, \rho^{(2)}, ..., \rho^{(m)}\}$ from which we wish to implicitly learn knowledge in service of deciding whether or not a query, α, is (approximately) $(1 - \epsilon)$-valid. Theorem 6 establishes that it's enough to possess a polynomial-time solver that is restriction-closed. Here, we observe that as long as the solver is sound and complete for $\mathcal{L}$, it will be restriction-closed. Thus, for many languages of interest, since we already know that we possess sound and complete solvers, we can immediately obtain implicit learning. Recall, formally:

Definition 7: [Sound and complete procedure] We say that A is a sound and complete decision procedure for language $\mathcal{L}$ if and only if, for any $\Delta, \alpha \in \mathcal{L}$, $\Delta \models \alpha$ if and only if $A(\Delta \wedge \neg\alpha)$=UNSAT.

Towards establishing that sound and complete solvers are restriction-closed, it will first be convenient to observe that the effect of restrictions is captured by adding conjunctive constraints.

Lemma 8: *Suppose* $\alpha \in \mathcal{L}$ *and* ρ *is a partial assignment. Then* $\alpha|_\rho$ *is satisfiable if and only if* $\alpha \wedge (\rho \downarrow)$ *is satisfiable.*

That is, for an assignment $\rho = \{x_1 = 1, x_2 = 0, x_3 = 1\}$ then $\rho \downarrow = (x_1 = 1 \wedge x_2 = 0 \wedge x_3 = 1)$.

We noted that completeness suffices to ensure restriction-closure. Sometimes it is not straightforward to show that the logic is restriction-closed (in particular, for more sophisticated SMT solvers, see in particular Beame et al. [2] for issues with CDCL solvers) however the guarantee that the method is complete becomes a sufficient condition for the restriction closure property:

Theorem 9: *Let* $\mathcal{L}$ *be a logical language such that for any* $\alpha \in \mathcal{L}$ *and any partial assignment* ρ *for* $\mathcal{L}$*,* $\alpha|_\rho \in \mathcal{L}$ *also. Let A be a sound and complete procedure for deciding entailment for* $\mathcal{L}$*. Suppose* $\Delta, \alpha \in \mathcal{L}$ *and* ρ *is a partial assignment for* $\mathcal{L}$*. If* $\Delta \models \alpha$*, then* $\Delta|_\rho \models \alpha|_\rho$*.*

Proof Suppose $\Delta \models \alpha$. By definition of soundness and completeness, $\Delta \models \alpha$ if and only if $A(\Delta \wedge \neg\alpha)$ = UNSAT. For any partial assignment ρ, we have that $A(\Delta \wedge \neg\alpha \wedge \rho\downarrow)$ = UNSAT. By the definition of a sound and complete decision procedure, $\Delta|_\rho \models \alpha|_\rho$ iff $A(\Delta|_\rho \wedge \neg\alpha|_\rho)$ = UNSAT, iff $A((\Delta \wedge \neg\alpha)|_\rho)$ = UNSAT, iff $A(\Delta \wedge \neg\alpha \wedge \rho\downarrow)$ = UNSAT. ∎

Thus, as a corollary of Theorem 6 together with Theorem 9, languages $\mathcal{L}$ with sound and complete solvers have solvers with implicit learning, that moreover are polynomial-time whenever the original solver was. Theorem 9 establishes that whenever the solver is complete for a restriction closed language, Algorithm 1 will be correct.

Algorithm 1 therefore represents the reduction from a learning to reason problem to a sound and complete solver by combining the theorem 6 and theorem 9. The rest of the results in the paper (Theorems 14, 16 , 17) illustrate the breadth of applicability of this style of analysis. In short, since we often have established the completeness of our solvers, Theorem 9 immediately establishes that we can add implicit learning.

Example 10: Consider a smart house system which maintains the rooms temperature and ventilation at the optimal condition. The system is capable of answering queries such as whether a particular room is sufficiently ventilated and in which case it will activate/turn on the fans for that room. The system is characterized by the following variables: { t (temperature in the room), CO (amount of carbon monoxide detected in the room), *nospeople* (the number of people present in the room), *vol* (volume of the room, depending on which current room is observed), *vent* (ventilation score which can be an integer between 1 and 5) }.

All these variables are constrained within some ranges and can be represented as a conjunction of constraints as an SMT formula. Now consider a knowledge base of the form: $KB = (15 < t < 32) \wedge (CO < 180) \wedge (0 \leq nospeople < 10) \wedge (0 < vent < 5)$. The system now observes values for some of the variables, say, the number of people in a room or the CO density. Because the observations are not full assignments of the state of the room at that specific moment, some of the variables are not seen. Consider that the system receives some partial interpretations of the form:

$\rho^{(1)} = \{nospeople = 3, vent = 3, t = 20, CO = *\}$

$\rho^{(2)} = \{nospeople = 4, vent = 3, t = 23, CO = *\}$

$\rho^{(3)} = \{nospeople = 5, vent = 3, t = 24, CO = *\}$

which does not contain any information about the carbon monoxide in that room, but we know from the knowledge base that the amount of CO is within some range. Consider answering the following query: $\alpha = (CO/nospeople) \leq 60$. Using the knowledge base alone is not sufficient to decide its entailment, but after receiving observations about how many people are in the room, we may assume an implicit knowledge base $I = nospeople \geq 3$. (That is, this formula is witnessed by the partial interpretations.) We are now able to decide that $KB \cup I \models \alpha$.

We reiterate that the KB alone is not sufficient to decide whether the query holds. Moreover, the partial assignments only give us information about some of the variables and so it is not sufficient to infer the truth of the query. We require both the KB and the partial assignments to establish that the query is entailed.

4 Difference Logic

Difference logic is a fragment of linear arithmetic where predicates are restricted to be of the form $x - y \bowtie k$, where x and y are variables from $\mathbb{Z}$ (or $\mathbb{R}$), k is a numeric constant, and $\bowtie \in \{<, >, \leq, \geq, =\}$. Nonetheless, the set of constraints will be assumed in the normal form $x - y \leq k$ by applying transformations for the other types of inequalities, as discussed in [23]. The fragment is useful for various verification problems involving timed automata, as well as for representing a class of probabilistic densities [3]. One can determine the satisfiability of difference constraints by viewing the constraints as a computational problem over a graph [16]. We discuss the essential ideas as they will help us understand why the restriction closure property holds, and obtain the learnability result. First, we discuss the construction of the *inequality graph* corresponding to a set of constraints:

Definition 11: [16] Let Δ be a set of difference predicates and let the inequality graph $G(V, E)$ be the graph comprising one edge (x_i, x_{i+1}) with weight k_i for every constraint of the form $x_i - x_{i+1} \leq k_i$ in Δ.

Given a difference logic formula Δ with non-strict inequalities only, the inequality graph corresponding to the set of difference predicates in Δ can be used for deciding entailment of a query α by refutation. A rephrasing of the needed result is:

Theorem 12: *[Adapted from [16]] Let $\Delta' = \Delta \wedge \neg\alpha$ be a conjunction of difference constraints and let G be the corresponding inequality graph. Then $\Delta' = \Delta \wedge \neg\alpha$ is UNSAT iff there is a negative cycle in G.*

The computation of the negative cycle proceeds by means of the Bellman-Ford algorithm, which solves single-source shortest-paths problem and is sound and complete. However note that, as mentioned earlier, we also need to be able to express restricted formulas in the language. For example, given $x - y \leq 4$ and on observing that y gets a value 2, we get that $x \leq 6$ which is *not* in the standard form for difference logic, so prima facie difference logic is *not* restriction-closed. But, we can consider an extension of the language by introducing a common integer variable ZERO that can be added to encode the predicate as $x - \text{ZERO} \leq 6$ [23]; when partial evaluation would introduce an inequality with a single variable, we encode it with the ZERO variable in this way. Using this extension of the language, we get the following result on restriction closure:

Proposition 13: *The constraint graph decision procedure is restriction closed.*

The restriction closure property then allows us to state the learnability result:

Corollary 14: *Let δ, ϵ, γ,m, Δ, α, $\rho^{(i)}$ be as introduced above. By utilizing the Bellman-Ford algorithm, Algorithm 1 returns Accept or Reject such that with probability at least $(1 - \delta)$:*

- *If there is some implicit KB I such that $\Delta \wedge I \models \alpha$ and every formula in I is witnessed true with probability $1 - \epsilon - \gamma$ under the partial models $\rho^{(i)}$, it returns Accept and*
- *If $\Delta \wedge \neg\alpha$ is $\epsilon + \gamma$-valid, it returns Reject.*

The algorithm runs in time $O(n(|\Delta| + |\alpha|)\frac{1}{2\gamma^2} \log \frac{1}{\delta})$, where n is the number of variables in Δ.

Proof The Bellman-Ford algorithm runs in time $O(|V||E|)$ [8], where V corresponds to the number of variables in Δ (so, n), and the set of edges is equivalent to the set of constraints plus the query; that is, $|E| = |\Delta| + |\alpha|$. Every iteration costs the time for checking feasibility[2] which is bounded by $O(n \times (|\Delta| + |\alpha|))$. The total number of iterations is $\frac{1}{2\gamma^2} \log \frac{1}{\delta}$ corresponding to the number of samples drawn, hence the total time bound is $O(n(|\Delta| + |\alpha|)\frac{1}{2\gamma^2} \log \frac{1}{\delta})$. ∎

We discuss the full fragment of linear arithmetic below but, of course, the benefit of the above result is that the Bellman-Ford algorithm is faster than any known algorithm for deciding the full fragment of linear arithmetic. Thus, if it suffices to use difference logic for the domain of interest, we should naturally restrict the reasoner to the constraint graph procedure.

5 Linear Arithmetic

Linear arithmetic is arguably one of the most important languages considered with SMT solvers. We assume formulas of the form $a_1x_1 + \ldots + a_nx_n \bowtie b$, where $x_1, \ldots, x_n$ are real (or integer) variables, $a_1, \ldots, a_n, b$ are rationals, and $\bowtie$ can be any of the inequalities, as introduced for difference logic. When $\bowtie$ ranges over the relations $\{=, \leq, \geq\}$, the satisfiability problem for conjunctions of such formulas is the standard *Linear Programming feasibility* problem. Numerous sound and complete polynomial-time algorithms exist for this problem. In particular, Cohen et al. [7] present a relatively efficient sound and complete algorithm. Assuming the program is given with rational number coefficients, the strict inequalities $\{<, >\}$ may be represented by adding a sufficiently small δ to nonstrict inequalities,

[2]The arithmetic operations are assumed to be performed in unit cost operation time, $O(1)$. This is in contrast to, for example, a model where integers are represented as strings.

using bounds on the size of the denominators obtained from Cramer's rule (a standard technique; see e.g., [10] for an effective implementation in practice). We note that partial evaluations continue to keep the program in the expected normal form. Thus, we get:

Proposition 15: *The linear programming decision procedure is restriction closed.*

For learnability, we therefore get:

Corollary 16: *Let $\delta, \gamma, m, \epsilon, \Delta, \alpha, \rho^{(i)}$ be as introduced above. Suppose we solve a linear program using the procedure from [7] which runs in time $O(n^{\omega+O(1)}\log(\frac{n}{\delta}))$, where n is the number of variables in the program and ω is the current matrix multiplication exponent ($\omega \approx 2.373$). Then Algorithm 1 using this decision procedure returns Accept or Reject such that with probability greater than $(1-\delta)$:*

- *If there is some implicit KB I such that $\Delta \wedge I \models \alpha$ and every formula in I is witnessed true with probability $1-\epsilon+\gamma$ under the partial models $\rho^{(i)}$, it returns Accept and*
- *If $\Delta \wedge \neg\alpha$ is $\epsilon+\gamma$-valid, it returns Reject.*

The algorithm runs in time $O(n^{\omega+O(1)}\log(\frac{n}{\delta})\frac{1}{2\gamma^2}\log\frac{1}{\delta})$.

Proof We use the algorithm of Cohen et al. for solving a system of inequalities, which runs in time $O(n^{\omega+O(1)}\log(\frac{n}{\delta}))$. For every iteration of the main algorithm, the system of inequalities is evaluated for some partial interpretation ρ, which is logically equivalent to evaluating whether $KB \wedge \rho \models \alpha \wedge \rho$. To decide entailment, we need determine whether the set of inequalities obtained from that statement all together offer a feasible solution. Every iteration costs the time for checking feasibility, bounded by $O(n^{\omega+O(1)}\log(\frac{n}{\delta}))$. The total number of iterations is $\frac{1}{2\gamma^2}\log\frac{1}{\delta}$, corresponding to the number of samples drawn, hence the total time bound is $O(n^{\omega+O(1)}\log(\frac{n}{\delta})\frac{1}{2\gamma^2}\log\frac{1}{\delta})$. ∎

6 Non-linear arithmetic

We now consider the more general case of "nonlinear" real arithmetic (NRA), i.e., of general semialgebraic sets, that is, systems of polynomial equations and inequalities. The canonical form of polynomial constraints is $p \bowtie 0$, where $\bowtie \in \{<, >, \leq, \geq\}$ and p is a sum of terms. Tarski [19] showed, using the method of quantifier elimination, that the first-order theory of the real numbers under addition and multiplication is decidable. As the result of plugging in some values for variables in to a polynomial is indeed another polynomial, and Tarski's algorithm is complete, we could apply our method to yield a solver based on Tarski's algorithm that implicitly learns such polynomial constraints:

Corollary 17: *(Implicit learnability over non-linear arithmetic) For a system of polynomial constraints Δ and a polynomial constraint α, Let δ, γ, M, m, and $\rho^{(i)}$ be as before. Then using Tarski's decision procedure [19] and $\frac{1}{2\gamma^2}\ln\frac{1}{\delta}$ partial assignments, with probability at least $(1-\delta)$, if α is not $(1-\epsilon)$-valid, Algorithm 1 outputs Reject, and if there is some implicit KB I such that $\Delta \wedge I \models \alpha$ and every formula in I is witnessed true under the partial models $\rho^{(i)}$ with probability at least $1-\epsilon+\gamma$, outputs Accept.*

However, the worst case time complexity of solving such first-order polynomial systems is doubly exponential in the number of variables [24], and in general Tarski's method would not be effective in practice. More recent algorithms have been proposed that appear to be much more effective in practice, e.g., Jovanovic and de Moura [11], in spite of the impossibility of a strong worst-case time complexity guarantee. The same argument holds for other fragments; for example, for the fragment of polynomial equalities. More effective methods are known that are sound and complete for such fragments, for example based on Grőbner bases [6, 5]: we can decide entailment by checking if the ideal generated by Δ remains the same when α is added by computing the Grőbner basis using Buchberger's algorithm. Although there is again no polynomial-time guarantee for such solvers – the problem is easily seen to be NP-hard – we can still apply our reduction to obtain an algorithm that implicitly learns polynomial equality constraints. Similarly, we note that for example, Tiwari and Lincoln [20] presented a solver that is effective for the instances arising in verification and synthesis, but is only complete for the fragment of polynomial equalities that have at most finitely many solutions. Since the number of solutions cannot increase when we substitute values for variables, we can apply our method to this fragment as well.

7 Discussion

We have motivated and proved learnability results for a number of fragments of SMT, by considering some of the main algorithmic schemes for the appropriate fragment. In this section, we briefly reflect the impact on learnability when considering other types of algorithmic schemes. A very popular and generic approach for solving SMT fragments is the *eager encoding* paradigm: an input formula in a non-propositional fragment can be encoded as a Boolean formula. This Boolean formula is equisatisfiable, but its effectiveness rests on a technique referred to as *bit-blasting* [1], which could result in a significant increase in the size of the resulting formula. In this case, we can resort to lifting the propositional learnability result from [12], but there is catch. A polynomial learnability result is only possible if the entailment question of Algorithm 1 can be solved in polynomial time, which we do not get for the full propositional fragment owing to the NP-completeness of propositional satisfiability. The approach taken in [12] is to "promise" that the query is provable in some low-complexity fragment; for example, it is provable by a small treelike resolution proof (where "small" refers to the number of lines of the proof). Equivalently, we give up on completeness, and only seek completeness with respect to conclusions provable in low complexity in a given fragment. In general, then, one obtains a running time guarantee that is parameterized by the size of the proof of the query. We can take a similar approach here, by using an algorithm for deciding entailment that is efficient when parameterized in such terms. In general, what is needed is a fragment for which we can decide the existence of proofs efficiently, and that is "restriction-closed," meaning that for any partial model, if we consider the restriction of each line of the proof, we obtain a proof in the same fragment. Most fragments we might consider, including specifically treelike or bounded-width resolution, are restriction-closed. So, while possible in principle if we consider the eager encoding paradigm, the caveat is that we have to rely on the low-complexity fragment capturing the reasoning problem in question.

Analogously, one could, in principle, appeal to the *lazy encoding* paradigm too [1]. Here, a Boolean *abstraction* of the input formula is considered, which consists of substituting all predicates over respective theories with fresh propositions. If a satisfying assignment is found, specialized theory solvers determine the validity of a proposed solution wrt the underlying theories. If such a procedure is sound and complete, then it would follow that it is restriction closed, but once again, the caveat would be that to obtain polynomial learnability, the Boolean reasoning has to be made tractable in the way discussed above. This might make the query answering process somewhat opaque, but it nonetheless would allow us to provide a learnability algorithm for a large class of SMT fragments [1]. We briefly note that Beame et al. [2] considered fragments of logics associated with CDCL solvers, and conjectured that these were the rare examples of logics that are *not* restriction-closed. Thus, using the techniques of Juba [12], it would appear that implicit learning cannot be added to CDCL solvers. But, since the solvers are complete, our approach here shows that the reduction is still correct for these solvers. The difference is that the running time, which corresponds to the size of the associated proof (extracted from the trace of the solver), could increase significantly.

8 Conclusion

We present new but also the first results on learning to reason with SMT. Our results show that for common fragments, such as difference logic and LRA, where sound and complete solvers are known, we can efficiently and robustly learn constraints when deciding entailment queries. Indeed, we showed more generally that for languages closed under substitutions of values for variables, including nonlinear arithmetic, implicit learning can always be added to sound and complete decision procedures. The main contributions are the extension of the framework to handle numeric data and Theorem 9 which establishes that we can add implicit learning to any complete solver.
We also considered alternative strategies such as eager encoding. One interesting direction for future work is to consider other kinds of partial information in our examples: Currently, these take the form of partial assignments. But, sometimes we might have domains in which sensors provide partial information of the form that some signal exceeds a detection threshold, or some value can be resolved to some interval (but no more precisely). We believe that these kinds of partial information can be

expressed naturally as additional LRA constraints, and thus we expect we should be able to use such partial examples in implicit learning. We would like a nice characterization of what implicit knowledge bases can be learned from such partial information and when.

Acknowledgements

Ionela G. Mocanu was supported by the Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training in Pervasive Parallelism (grant EP/L01503X/1) at the University of Edinburgh, School of Informatics. Vaishak Belle was supported by a Royal Society University Research Fellowship. Brendan Juba was supported by NSF Award CCF-1718380. We would also like to thank our reviewers for their helpful suggestions.

References

[1] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. *Satisfiability modulo theories*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 825–885. 1 edition, 2009.

[2] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.*, 22:319–351, 2004.

[3] V. Belle, A. Passerini, and G. Van den Broeck. Component caching in hybrid domains with piecewise polynomial densities. In *AAAI*, 2016.

[4] Vaishak Belle and Brendan Juba. Implicitly learning to reason in first-order logic. *CoRR*, abs/1906.10106, 2019.

[5] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basis-elemente des Restklassenrings nach einem nulldimensionalen Polynomideal*. PhD thesis, 1965.

[6] Bruno Buchberger. Groebner bases: An algorithmic method in polynomial ideal theory. In N.K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publ. Comp., 1985.

[7] Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *CoRR*, abs/1810.07896, 2018.

[8] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[9] Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. In *COLT*, pages 815–830, 2016.

[10] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for dpll(t). In *Proceedings of the 18th International Conference on Computer Aided Verification*, CAV'06, pages 81–94, Berlin, Heidelberg, 2006. Springer-Verlag.

[11] Dejan Jovanović and Leonardo De Moura. Solving non-linear arithmetic. In *International Joint Conference on Automated Reasoning*, pages 339–354. Springer, 2012.

[12] Brendan Juba. Implicit learning of common sense for reasoning. In *IJCAI*, pages 939–946, 2013.

[13] Michael J Kearns, Robert E Schapire, and Linda M Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

[14] Roni Khardon and Dan Roth. Learning to reason. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, pages 682–687. AAAI Press, 1994.

[15] Samuel Kolb, Stefano Teso, Andrea Passerini, and Luc De Raedt. Learning SMT(LRA) constraints using SMT solvers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 2333–2340. AAAI Press, 2018.

[16] Daniel Kroening and Ofer Strichman. *Linear Arithmetic*, page 111–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[17] Loizos Michael. Partial observability and learnability. *Artificial Intelligence*, 174(11):639–669, 2010.

[18] Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994. Special Issue: Ten Years of Logic Programming.

[19] Alfred Tarski. A decision method for elementary algebra and geometry. *Journal of Symbolic Logic*, 17(3):207–207, 1952.

[20] Ashish Tiwari and Patrick Lincoln. A nonlinear real arithmetic fragment. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification*, page 729–736, Cham, 2014. Springer International Publishing.

[21] Leslie G. Valiant. Robust logics. *Artificial Intelligence*, 117(2):231–253, 2000.

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

[23] Chao Wang, Franjo Ivančić, Malay Ganai, and Aarti Gupta. Deciding separation logic formulae by sat and incremental negative cycle elimination. In Geoff Sutcliffe and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 322–336, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[24] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1):3–27, 1988.