On the Complexity of Some Inductive Logic Programming Problems *

Georg Gottlob, Nicola Leone, Francesco Scarcello

Institut für Informationssysteme
Technische Universität Wien
Paniglgasse 16, A-1040 Wien, Austria;
Internet: gottlob@dbai.tuwien.ac.at

Abstract. The bounded ILP-consistency problem for function-free Horn clauses is described as follows. Given a set E^+ and E^- of function-free ground Horn clauses and an integer k polynomial in $E^+ \cup E^-$, does there exist a function-free Horn clause C with no more than k literals such that C subsumes each element in E^+ and C does not subsume any element in E^-. It is shown that this problem is Σ_2^P complete. We derive some related results on the complexity of ILP and discuss the usefulness of such complexity results.

1 Introduction

The complexity analysis of reasoning problems in various areas such as non-monotonic reasoning (NMR), logic programming (LP), machine learning (ML), and inductive logic programming (ILP) has attracted much attention and interest in the last years. It was shown that several important reasoning problems are not only intractable, i.e. **NP** hard, but reside at higher levels of the polynomial hierarchy. In particular, it turned out that a large number of problems in NMR and LP reside at the *second level of the polynomial hierarchy* and are complete for either Σ_2^P or the dual class Π_2^P.

In the present paper we deal with the *bounded ILP-consistency problem for function-free Horn clauses*, described as follows. Given a set E^+ and E^- of function-free ground Horn clauses and an integer k polynomial in $E^+ \cup E^-$, does there exist a clause C having at most k literals such that C subsumes each element in E^+ and C does not subsume any element in E^-? We show that this problem is Σ_2^P complete. We also derive some results on the complexity of related ILP problems.

A proof that a given reasoning problem A is complete for a particular complexity class in the polynomial hierarchy, say, Σ_2^P, provides us with a deep qualitative understanding of the problem. For example, we can "extract" the following benefits from such a proof or result:

* This work has been supported by *FWF (Austrian Science Funds)* under the project P11580-MAT "A Query System for Disjunctive Deductive Databases" and by the *ISI-CNR, Istituto per la Sistemistica e l'Informatica (Italian National Research Council)*, under grant n.224.07.5.

- The proof allows us to recognise the sources of complexity in the problem and thus provides a systematic way of finding tractable subcases (by eliminating all these sources). For example, if a problem is **NP**-complete, then it contains basically one source of intractability, related to a choice to be made among exponentially many candidates. If a problem is Σ_2^P-complete then there are usually two intermingled sources of complexity. One is, as in **NP**, a choice problem, and the other one is a checking problem (checking whether the choice was a good choice) which is co-**NP** hard and depends on this choice. In order to prove a problem A Σ_2^P hard, we usually reduce a well-known other Σ_2^P hard problem B to A. Assuming that the sources of complexity of B are identified, the sources of complexity of A become clear via the translation.
- The complexity result enables us to classify the problem and helps identifying other problems in the same complexity class to which the problem is polynomially translatable. For these other problems practical solution algorithms may already exist which we can take profit of. For example, in this paper we show that a relevant ILP problem, the *bounded Horn clause consistency problem* is complete for the complexity class Σ_2^P. On the other hand, it was shown that the relevant decision problems in disjunctive logic programming (DLP) and in Reiter's default logic [28] are also Σ_2^P complete [16, 9, 11]. It follows that our ILP problem can be polynomially translated into equivalent DLP or default logic problems and solved by sophisticated existing software packages for DLP or default logic [8, 10, 3]. In such a way one can profit from the highly efficient data management and search pruning techniques implemented in these systems. Note that such translations can be given for the bounded Horn clause ILP problem as well, i.e., for the *search* problem of *finding* a Horn clause C with no more than k literals that subsumes E^+ and does not subsume any clause in E^-. We are currently investigating and experimenting implementations of ILP problems on top of the DLP system described in [10].
- The complexity result enables us to establish negative translation results as well. For example, there are several forms of nondisjunctive logic programming i.e., LP with negation in rule bodies, but without disjunction in rule heads. The strongest form of nondisjunctive logic programming for which practical interpreters were constructed is Logic Programming under the *stable model semantics* [13]. It was shown that this form of logic programming is **NP** complete [21]. Therefore, unless the Polynomial Hierarchy collapses, it is not the case that the bounded Horn clause ILP-consistency problem can be polynomially translated to a nondisjunctive logic programming problem. Of course also such "negative" results are very useful. They prevent researchers or programmers to waste their precious time attempting to effectively solve a problem using techniques, programming languages, or formalisms that are not enough powerful.
- The theory of molecular computing envisages the possibility of nonelectronic devices based on natural phenomena that overcome classical intractability

by exploiting massive parallelism. In [1], DNA strings are used as elementary processing units. **NP** hard problems can be solved in polynomial time if a sufficient (exponential) number of DNA strings with certain properties is provided and if certain elementary operations are assumed to be feasible. A basic model of DNA computing [1] was formally defined and studied in [20]. The complexity of this method and of some relevant extensions is studied in [31]. In [31] it was recently proved that the basic model of DNA computing of [20], can be used to solve precisely the problems in the complexity class $\Delta_2^P = \mathbf{P}^{\mathbf{NP}}$ in polynomial time, while a second, slightly more powerful model, is able to solve precisely all problems in the class $\Delta_3^P = \mathbf{P}^{\Sigma_2^P}$ in polynomial time. Thus, once we have shown that a problem is in Σ_2^P, it is solvable in polynomial time by a DNA computer. Note that due to such results on DNA computers, the exact location of a problem in the Polynomial Hierarchy becomes yet more interesting. In fact, a **PSPACE** complete problem would most likely not be solvable in polynomial time by the above quoted models of DNA computers, while a Σ_2^P complete problem can be solved in polynomial time by the second, but probably not by the first model. On the other hand, any **NP** complete problem is solvable by DNA computers of either type. The future will show which type of DNA computers can be realized in practice, and whether such computers will exist at all.

In summary, the complexity analysis of a problem gives us much more than merely a quantitative statement about its tractability or intractability in the worst case. Rather, locating a problem at the right level in the polynomial hierarchy gives us a deep *qualitative* knowledge about this problem which – in many cases – can be exploited for applications.

2 Preliminaries and Previous Results

2.1 Complexity Theory

We assume that the reader has some background on the concept of **NP**-completeness [12, 17]. Upon the class **NP**, the polynomial hierarchy (PH) has been defined as a subrecursive analog to the Kleene arithmetical hierarchy. For any complexity class C, let $\mathbf{NP}^C$ denote the decision problems solvable in polynomial time by some nondeterministic oracle Turing machine with an oracle for any problem in C. The classes Δ_k^P, Σ_k^P, and Π_k^P of PH are defined as follows:

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathbf{P}$$

and for all $k \geq 0$,

$$\Delta_{k+1}^P = \mathbf{P}^{\Sigma_k^P}, \quad \Sigma_{k+1}^P = \mathbf{NP}^{\Sigma_k^P}, \quad \Pi_{k+1}^P = \mathrm{co}\Sigma_{k+1}^P.$$

In particular, $\mathbf{NP} = \Sigma_1^P$, co-$\mathbf{NP} = \Pi_1^P$, $\Sigma_2^P = \mathbf{NP}^{\mathbf{NP}}$, and $\Pi_2^P = \text{co-}\mathbf{NP}^{\mathbf{NP}}$. The classical **NP**-complete problem is to decide if a collection

$\mathcal{C} = \{L_{i,1} \vee \cdots \vee L_{i,n_i} : 1 \leq i \leq m\}$ of propositional clauses is simultaneously satisfiable (SAT). This problem is still **NP**-complete if each clause $L_{i,1} \vee \cdots \vee L_{i,n_i}$ in $\mathcal{C}$ contains only positive literals or only negative literals (MSAT) [12]. The most prominent Σ_2^P-complete problem is to decide the validity of a formula from $\mathrm{QBF}_{2,\exists}$, the set of quantified Boolean formulae of form $\exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m E$, where $E = E(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a propositional formula built from atoms $x_1, \ldots, x_n, y_1, \ldots, y_m$.

2.2 Inductive Logic Programming

The computational complexity of learning concepts from examples or from particular kind of queries to an hypothetical teacher, has been deeply investigated in many papers from different research areas (see, for instance, [14, 32, 19, 18, 6, 4, 5, 7]), ranging from computational learning theory to artificial intelligence.

In this paper we focus on some complexity problems arising in inductive logic programming (ILP) [22, 23, 24]. This relatively new field is concerned with the induction of concepts expressed in clausal form (typically first order Horn clauses) from examples and background knowledge. Compared with previous frameworks, ILP should result more expressive, because first order theories are allowed instead of only propositional formulae. Furthermore, the introduction of background knowledge can be useful for a simpler modeling of some learning tasks.

We next describe formally the problems addressed by ILP. Our formalization is similar to the presentation in [18].

Let $\mathcal{L}$ denote the language of first order logic. An *ILP setting* is a tuple $\mathcal{S} = \langle \mathcal{L}_B, \mathcal{L}_H, \vdash, L_E \rangle$, where:

- $\mathcal{L}_B \cup \mathcal{L}_H \cup \mathcal{L}_E \subseteq \mathcal{L}$, and
- $\vdash$ is a correct provability relation for the language $\mathcal{L}$.

The *ILP-learning* problem for an ILP setting $\mathcal{S}$ is:

Given $\langle B, EX^+, EX^- \rangle$, with $B \in \mathcal{L}_B$ (background Knowledge), $EX^+ \in \mathcal{L}_E$ (positive examples), and $EX^- \in \mathcal{L}_E$ (negative examples), find an hypothesis $H \in \mathcal{L}_H$ such that: (i) $H \wedge B \vdash ex \ \forall ex \in EX^+$, (ii) $H \wedge B \nvdash ex \ \forall ex \in EX^-$, and (iii) $B \wedge H \nvdash \square$ (i.e., H is consistent with B).

The *ILP-consistency* problem for an ILP setting $\mathcal{S}$ is the problem of deciding whether a solution for the *ILP-learning* problem does exist or not.

It is worth noting that solving the ILP problem is not sufficient in general for a correct learning of the unknown concept from which the given examples are taken. To be sure that our hypothesis H coincides with the unknown concept (or it is at least a good approximation) we need some learnability model which provides sufficient conditions for a correct identification of the concept.

One of the most important and well studied approach is Valiant's Probably-Approximately-Correct (PAC) learning. PAC learning is a probabilistic distribution free approach to learning. There exists a fixed but unknown probability

distribution D according to which examples are drawn and against which the quality of approximation is evaluated.

Following [18], we next define when an ILP setting is PAC learnable.

Definition 1. An ILP setting $\mathcal{S} = \langle \mathcal{L}_B, \mathcal{L}_H, \vdash, L_E \rangle$ is (polynomially) PAC learnable, if and only if there exists an algorithm PLEARN and a polynomial function $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_c, n_b)$, so that for every $n_e > 0$, every $n_b > 0$, every background knowledge $B \in \mathcal{L}_B$ of size n_b or less, every concept $C \in \mathcal{L}_H$ with size n_c or less, every $\epsilon : 0 < \epsilon < 1$, every $\delta : 0 < \delta < 1$, and every probability distribution D, for any set of examples $E = E^+ \cup E^-$ of C, drawn from $\mathcal{L}_E$ according to D and containing at least $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_c, n_b)$ examples

1. PLEARN, on input E, B, ϵ, and δ outputs a hypothesis $H \in \mathcal{L}_H$ such that

$$P(D(Cover(H) \ \Delta \ Cover(C)) > \epsilon) < \delta,$$

   where $Cover(X) = \{e \in \mathcal{L}_E \mid X, B \vdash e\}$ and Δ denotes the simmetric set difference;
2. PLEARN runs in time polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_c, n_b$, and the number of examples, and
3. for all $e \in \mathcal{L}_E$, the truth of the statement $e \in Cover(H)$ can be tested in polynomial time.

The polynomial function $m(\frac{1}{\epsilon}, \frac{1}{\delta}, n_e, n_c, n_b)$ is called the sample complexity of the algorithm PLEARN.

A very frequently used provability relation is ϑ-subsumption [29, 27]:

Definition 2. A clause C *ϑ-subsumes* a clause E ($C \vdash_\vartheta E$), iff there exist a substitution ϑ such that $C\vartheta \subseteq E$. If S is a set of clauses, then $C \vdash_\vartheta S$ if C subsumes all clauses in S.

2.3 Complexity Results in ILP

In a recent interesting paper [18], Kietz and Džeroski proved that a restricted class of function free clauses (i.e. clauses without function symbols, including constants), namely k-discriminative non recursive ij-determinate predicate definitions, are PAC learnable.

They also considered whether it is possible to remove some syntactical restriction without loosing the polynomial-time learnability. The answer was negative, since the consistency problem for $\mathcal{L}_H = \{i2\text{-determinate Horn clauses}\}$ turned out to be PSPACE-hard, if i is variable.

Furthermore, a weaker restriction on the way variables in a clause can be linked has been analysed.

Definition 3 [18]. A Horn clause is linked if all of its literals are linked. A literal is linked if at least one of its terms is linked. A term is linked with a linking-chain of length 0 if it occurs in the head of the clause. A term in a literal is linked

with a linking-chain of length $d+1$, if another term in the same literal is linked with a linking-chain of length d. The depth of a term is the minimal length of its linking-chains. A linked Horn clause which has maximal depth of terms i and maximal arity of literals j, is called i,j-nondeterminate.

In [18] it was also shown that the consistency problem for the ILP setting

$\langle \emptyset$, 1,2-nondeterminate function-free Horn clauses, $\vdash_\vartheta$, ground Horn clauses$\rangle$

is **NP**-hard. As a consequence, such a setting is not PAC-learnable as long as the widely assumed **RP**$\neq$ **NP** conjecture is true.

Other important results are contained in the work [6] by Cohen and Page, that give an overview of the most useful techniques for deriving either positive or negative results in the PAC-learning and learning from equivalence queries frameworks.

In the latter learning model, the learner receives information about the unknown concept by asking *equivalence queries*. An equivalence query is a concept in the hypothesis language which is sent to an oracle that replies "equivalent" if the given concept is equivalent to the unknown concept, or it returns a counterexample. If an ILP setting $\mathcal{S}$ is learnable from equivalence queries, then $\mathcal{S}$ is also PAC-learnable.

As it was pointed out [6, 26], a non-learnability result for a setting $\mathcal{S}$ based on a consistency-hardness result does not entail the non-learnability of some setting more expressive than $\mathcal{S}$.

A stronger result can be obtained by an evaluation hardness argument, i.e. by showing that there exists a concept in the language whose evaluation is not feasible in polynomial time. *Evaluation*, in this context means deciding whether an example is or is not implied (via the respective $\vdash_\vartheta$ relation) by a hypothesis. Indeed, if a setting is PAC-learnable, then it should always be possible to find an hypothesis which is consistent with the given examples and evaluable in polynomial time [30].

Along these lines, Cohen and Page [6] proved that the ILP setting $\mathcal{S}_{k,A}$ = $\langle$ground facts of
bounded arity$A$, k-linked Horn clauses, $\models$, ground facts$\rangle$, for any fixed $k \geq 1$ and any fixed $A \geq 3$ is not PAC-learnable, and neither is any setting at least as expressive as $\mathcal{S}_{k,A}$.

They also found some positive results. For instance, it turns out that a setting strictly more expressive than the one based on ij-determinate non recursive clauses (as hypothesis language) is PAC-learnable.

3 The Bounded Horn Clause ILP-Consistency Problem

The ILP-consistency problem can be very hard or even undecidable even for very simple logic programming settings.

However, one is not always interested in finding *arbitrarily large* solutions. Rather, it makes sense to impose an upper bound on the size of the desired

solutions. It makes further sense to require that this bound be polynomially related to the size of the supplied example and counterexample sets.

Definition 3.1 *The* bounded ILP-learning *problem for an ILP setting $\mathcal{S}$ is:*

Given $\langle B, EX^+, EX^- \rangle$ and an integer k, with k polynomially bounded in the size of $\langle B, EX^+, EX^- \rangle$, $B \in \mathcal{L}_B$, $EX^+ \in \mathcal{L}_E$, $EX^- \in \mathcal{L}_E$, find an hypothesis $H \in \mathcal{L}_H$ with at most k literals, such that: (i) $H \wedge B \vdash ex \ \ \forall ex \in EX^+$, and (ii) $H \wedge B \not\vdash ex \ \ \forall ex \in EX^-$, and (iii) $B \wedge H \not\vdash \square$ (i.e., H is consistent with B).

The bounded ILP-consistency *problem for an ILP setting $\mathcal{S}$ is accordingly defined.*

The NP-completeness of the ϑ-subsumption checking is a well-known result [12, 2]. Nevertheless, we state a lemma about the complexity of the complementary problem, whose proof will be useful for a better understanding of the theorem below.

Observe that, if ϑ-subsumption is adopted as the provability relation of the ILP setting, then Condition (iii) of the definition of the ILP learning problem (both in general and bounded version) is not relevant, as ϑ-subsumption guarantees consistency. Therefore, in the sequel of the paper, we will not consider this condition for ILP settings based on ϑ-subsumption.

Lemma 4 (Baxter [2]). *Given a function-free Horn clause C and a ground Horn clause P, deciding whether $C \not\vdash_\vartheta P$ is co-**NP**-complete.*

PROOF. *Membership in co-**NP**.* We decide whether $C \vdash_\vartheta P$ (the complementary problem of $C \not\vdash_\vartheta P$) in **NP** as follows. Guess a substitution ϑ for C, check that $C\vartheta \subseteq P$. Since the size of a substitution for C is polynomial in the size of C, and the containment check can be done in polynomial time, the problem lies in **NP**. As a consequence, deciding whether $C \not\vdash_\vartheta P$ is in co-**NP**.

*co-**NP**-Hardness.* We transform the (co-**NP**-hard) problem of deciding whether a QBF $\Psi = \forall ZE$ is in $\mathrm{QBF}_{1,\forall}$, where E is in 3DNF, into deciding whether a horn clause C does not subsume a ground Horn clause P.

Let $Z = \{z_1, \ldots, z_n\}$, and $E = E_1 \vee \cdots \vee E_t$.

To clarify the transformation, we will use QBF $\Psi = \forall z_1, z_2, z_3, z_4 E$, where $E = (z_1 \wedge \neg z_2 \wedge z_3) \vee (\neg z_1 \wedge z_2 \wedge z_4)$, as a running example.

Let z_1^i, z_2^i, z_3^i denote the (positive) boolean variables of E_i. A falsifying scheme for E_i is a triple $f = \langle b_1, b_2, b_3 \rangle$ of boolean values such that assigning b_k to z_k^i $(1 \leq i \leq 3)$ makes E_i false. We denote by $fs(E_i)$ the set of the falsifying schemes for E_i.

For instance, in our running example, $\langle 0,0,0 \rangle$, $\langle 0,1,0 \rangle$, and $\langle 1,1,1 \rangle$ are some falsifying schemes for $E_1 = z_1 \wedge \neg z_2 \wedge z_3$. Falsifying schemes represent the boolean assignments to the variables of E_i that do not satisfy E_i; therefore, every conjunct E_i has exactly 7 falsifying schemes.

Consider the following ground Horn clause $P(E)$:

$$P(E) = \leftarrow \bigwedge_{1 \leq i \leq t} \bigwedge_{f \in fs(E_i)} p_i(f).$$

For the running example, we will build the following ground horn formula including all the falsifying schemes for the two conjuncts:

$$\begin{aligned} P(E) = \leftarrow\ & p_1(0,1,1) \wedge p_1(0,0,1) \wedge p_1(0,1,0) \wedge p_1(1,1,0) \wedge \\ & p_1(1,1,1) \wedge p_1(0,0,0) \wedge p_1(1,0,0) \wedge \\ & p_2(1,0,1) \wedge p_2(1,0,0) \wedge p_2(1,1,1) \wedge p_2(1,1,0) \wedge \\ & p_2(0,0,1) \wedge p_2(0,0,0) \wedge p_2(0,1,0) \end{aligned}$$

Moreover, let $C(E)$ be the following Horn clause:

$$C(E) = \leftarrow \bigwedge_{1 \leq i \leq t} p_i(Z_1^i, Z_2^i, Z_3^i).$$

In our example, we get

$$C(E) = \leftarrow p_1(Z_1, Z_2, Z_3) \wedge p_2(Z_1, Z_2, Z_4).$$

The following equivalence holds:

$$C(E) \not\vdash_\vartheta P(E) \Longleftrightarrow \Psi \in \mathrm{QBF}_{1,\forall}$$

Indeed, the substitutions that can prove the subsumption $C(E) \vdash_\vartheta P(E)$ map necessarily the Z_j variables to $\{0,1\}$ and therefore correspond to boolean truth assignment to the variables z_j $(1 \leq j \leq n)$ of Ψ. In particular, given a substitution ϑ, $C(E)\vartheta \subseteq P(E)$ holds if and only if ϑ represents a truth assignment that falsifies E. Therefore, there exists no truth assignment falsifying E (i.e., $\Psi \in \mathrm{QBF}_{1,\forall}$) if and only if there exists no substitution ϑ such that $C(E)\vartheta \subseteq P(E)$ (i.e., $C(E) \not\vdash_\vartheta P(E)$).

For instance, we can easily recognise that the formula Ψ of the running example is not valid. Indeed, let ϑ be the following substitution for the variables in $C(E)$: $\{Z_1 = 1, Z_2 = 1, Z_3 = 0, Z_4 = 0\}$. Then, $C(E)\vartheta \subseteq P(E)$, since both $p_1(1,1,0)$ and $p_2(1,1,0)$ belong to $P(E)$. Hence, $C(E) \vdash_\vartheta P(E)$. □

Let $\mathcal{S}_{HC}$ be the setting $\langle \emptyset$, function-free Horn clauses, $\vdash_\vartheta$, ground Horn clauses$\rangle$. Then, the following theorem holds.

Theorem 5. *The bounded ILP-consistency problem for $\mathcal{S}_{HC}$ is Σ_2^P-complete.*

PROOF. *Membership in Σ_2^P.* Let a set EX^+ of positive examples, EX^- of negative examples and an integer k be given. From the definition of $\mathcal{S}_{HC}$, EX^+ and EX^- are ground Horn clauses. If the set of positive examples EX^+ is empty, then the problem is trivial, as the clause $\leftarrow q$, where q is a predicate which does not appear in EX^- is a solution.

Assume now that EX^+ is not empty and let $Pred(EX^+)$ be the set of predicates (symbols) appearing in EX^+. Then, every possible witness of the consistency, that is, every clause C that subsumes the positive examples (and does not subsume the negative ones), will contain *only* predicates from $Pred(EX^+)$. Therefore, we can decide the bounded ILP consistency problem at hand as follows. Guess a clause C with a number of atoms less than or equal to k whose predicates are taken from $Pred(EX^+)$, and check the following: (i) for each clause P in EX^+, verify that $C \vdash_\vartheta P$; (ii) for each clause P' in EX^-, verify that $C \nvdash_\vartheta P'$.

The size of C is polynomial in the input, as k is polynomially bounded, and each predicate in C is present in the input. Moreover, from Lemma 4, checking whether $C \nvdash_\vartheta P'$ can be done in co-**NP** (and checking that $C \vdash_\vartheta P$ can be done in **NP**). Therefore, by a polynomial number of calls to an **NP** oracle, we are able to check that the guessed clause C is indeed a solution. The bounded ILP-consistency problem for $\mathcal{S}_{HC}$ is hence in Σ_2^P.

Σ_2^P-Hardness. We transform deciding that a QBF $\Phi = \exists X \forall Y E$ is in $\mathrm{QBF}_{2,\exists}$, where E is a 3DNF, into deciding whether there exists a function-free Horn clause C, with at most k literals, such that $C \vdash_\vartheta EX^+$ and $C \nvdash_\vartheta ex \quad \forall ex \in EX^-$, where k, EX^+ and EX^- are given in input.

Without loss of generality, we assume that Φ does not contain useless variables or redundant or trivially unsatisfiable conjuncts. More precisely, we assume that (i) the same boolean variable does not occur twice in the same conjunct of E; (ii) each boolean variable in X appears at least in one conjunct of E; (iii) there exists no conjunct of E containing only Y-variables not appearing in any other conjunct of E; and (iv) each conjunct E_i contains exactly 3 boolean variables.

To clarify the transformation, we will use QBF

$$\Phi_0 = \exists x_1, x_2 \forall y_1, y_2 (x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_2 \wedge y_2)$$

as a running example.

We next specify k, EX^+ and EX^-. Let $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_n\}$, $E = E_1 \vee \cdots \vee E_t$, and let z_1^i, z_2^i, z_3^i denote the (positive) boolean variables of a conjunct E_i. Moreover, let $0, 1, c, ch, cx_1, \cdots, cx_m, cy_1, \cdots, cy_n, cx'_1, \cdots, cx'_m$, be distinct constants.

For the QBF Φ, we fix $k = t + m$, that is we look for function-free Horn clauses having at most k atoms, where k is the number of conjunct in Φ plus the number of X variables (i.e., the number of the variables existentially quantified).

We have only one positive example, which we call *master example*:

$$ex1^+ = \leftarrow \bigwedge_{1 \le i \le t} p_i(const(z_1^i), const(z_2^i), const(z_3^i)) \\ \bigwedge_{1 \le j \le m} val_j(cx_j, cx'_j) \bigwedge_{1 \le j \le m} val_j(cx'_j, cx_j)$$

where

$$const(z) = \begin{cases} cx_k \text{ if } z \text{ is a } X \text{ variable } x_k \\ cy_k \text{ if } z \text{ is a } Y \text{ variable } y_k \end{cases}$$

For the sample QBF Φ_0, the *master example* $ex1^+(\Phi_0)$ is the following (ground Horn) clause:

$$\leftarrow p_1(cx_1, cx_2, cy_1) \wedge p_2(cx_1, cx_2, cy_2) \wedge \\ val_1(cx_1, cx_1') \wedge val_2(cx_2, cx_2') \wedge val_1(cx_1', cx_1) \wedge val_2(cx_2', cx_2)$$

We next define the negative examples. We call *enforcing examples*, the following negative examples:

($n1$) For each predicate p_i $(1 \leq i \leq t)$, $ex1^-_{p_i}$ is the clause obtained from $ex1^+$ by replacing the (only) atom with predicate p_i by $p_i(c, c, c)$.

($n2$) For each predicate val_j $(1 \leq i \leq m)$, $ex1^-_{val_j}$ is the clause obtained from $ex1^+$ by replacing both the atoms with predicate val_j by $val_j(c, c)$ (i.e., $val_j(ch, cx_j) \wedge val_j(ch, cx_j')$ is replaced by $val_j(c, c)$).

We denote the set of *enforcing examples* by $EX1^-$.

For the running example, the enforcing example $ex1^-_{p_1}$ and $ex1^-_{val_1}$ are the following ground horn clauses:

$$\leftarrow p_1(c, c, c) \wedge p_2(cx_1, cx_2, cy_2) \wedge \\ val_1(cx_1, cx_1') \wedge val_2(cx_2, cx_2') \wedge val_1(cx_1', cx_1) \wedge val_2(cx_2', cx_2)$$

$$\leftarrow p_1(cx_1, cx_2, cy_1) \wedge p_2(cx_1, cx_2, cy_2) \wedge \\ val_1(c, c) \wedge val_2(cx_2, cx_2') \wedge val_2(cx_2', cx_2)$$

Finally, we define the negative example ex_2^-, which we call *checking example*:

$$ex_2^- = P(E) \bigwedge_{1 \leq j \leq m} val_j(1, 0)$$

where $P(E)$ is the ground clause specified in the proof of Lemma 4.

In the running example, ex_2^- is the following ground Horn clause:

$$\leftarrow p_1(0,1,1) \wedge p_1(0,0,1) \wedge p_1(0,1,0) \wedge p_1(1,1,0) \wedge \\ p_1(1,1,1) \wedge p_1(0,0,0) \wedge p_1(1,0,0) \wedge \\ p_2(1,0,1) \wedge p_2(1,0,0) \wedge p_2(1,1,1) \wedge p_2(1,1,0) \wedge \\ p_2(0,0,1) \wedge p_2(0,0,0) \wedge p_2(0,1,0) \wedge \\ val_1(1,0) \wedge val_2(1,0)$$

We claim that the QBF Φ is valid if and only if there exists a function-free Horn clause C, with at most k literals, such that C subsumes $ex1^+$ and C does not subsume any example in $EX1^- \cup \{ex2^-\}$.

Only if part. Assume Φ is valid. Then, there exists an assignment T to the boolean variable belonging to the set X such that, for every assignment to the variable in the set Y, the 3DNF formula E is satisfied.

Now, consider the following function free Horn clause:

$$C: \quad \leftarrow \bigwedge_{1 \leq i \leq t} p_i(Z_1^i, Z_2^i, Z_3^i) \\ \bigwedge_{1 \leq j \leq m} val_j(X_j', X_j'')$$

where we have

(i) Z_j^i is the (X or Y) variable corresponding to the boolean variable appearing in the j-th position of the conjunct E_i (e.g., for $E_i = x_1 \wedge \neg x_2 \wedge y_1$, we have $p_i(X_1, X_2, Y_1)$);
(ii) if $T(x_j) = true$, then X'_j is the variable X_j and X''_j is a new variable (independent from all other variables); otherwise ($T(x_j) = false$), X''_j is the variable X_j and X'_j is a new variable.

C subsumes the master example and cannot subsume any of the enforcing examples. Furthermore, if the val_j atoms in C can be mapped into all the val_j ground atoms in the checking example, then each X variable X_j must be bound to value 1 if $T(x_j) = true$ and to value 0 if $T(x_j) = false$. Indeed, in the checking example, we have $val_j(1,0)$ (for each $j \in \{1, \ldots, m\}$). Then if variable X_j appears in the first place of the atom with predicate val_j (i.e., $val_j(X_j, X''_j) \in C$), it should be set to value 1; otherwise (i.e., $val_j(X'_j, X_j) \in C$), it is forced to unify with constant 0.

Therefore, the assignment to the X variables is encoded in the clause C by means of the positions of each variable in the corresponding val_j atom. When we check the validity of E by verifying whether $C \vdash_\vartheta ex2^-$, the values of X variables are all fixed according to the assignment T. Hence, following Lemma 4, C cannot subsume the checking example, or it should exist an assignment to the Y variables which falsifies the formula E.

If part. Suppose there exists a function free Horn clause C with at most k literals which subsumes the master example and does neither subsume any enforcing example nor the checking example. We prove that, in such a case, Φ is valid.

Fact a). Since $C \vdash_\vartheta ex1^+$, we know all predicates in C are among those used in the master example $ex1^+$.

Fact b). In the master example we have a different constant for each variable we want to be distinct. In particular, we have a different constant for each boolean variable appearing in the formula E. Thus, each variable in C corresponds to only one boolean variable appearing in the formula E. Furthermore, the two variables appearing in a val_j atom are distinct.

Fact c). From the enforcing examples, we know at least one atom for each predicate used in the master example must belong to C. Indeed, assume there is no atom in C with a predicate p_i. Then, C should subsume the enforcing example $ex1^-_{p_i}$, by applying the same substition used to subsume the master example. The same clearly holds for the val_j predicates.

Fact d). From fact c), and from the hypothesis C contains at most k literals, we can conclude C contain exactly one atom for each predicate appearing in the master example. In particular, only one atom for each predicate val_j can appear in C.

Let ϑ be a substitution such that $C\vartheta \subseteq ex1^+$ (such a ϑ exists by hypothesis) and let $\bar{\vartheta}$ be the (non ground) substitution which acts as follows on a variable Z appearing in C:

- if $\vartheta(Z) = cx_j$ then $\bar{\vartheta}(Z) = X_j$;
- if $\vartheta(Z) = cx'_j$ then $\bar{\vartheta}(Z) = X'_j$;
- if $\vartheta(Z) = cy_i$ then $\bar{\vartheta}(Z) = Y_i$;

where $1 \leq i \leq n$ and $1 \leq j \leq m$. Let $\bar{C}$ be the function free Horn clause obtained by applying the substitution $\bar{\vartheta}$ to the clause C, i.e. $C\bar{\vartheta} = \bar{C}$. It is easy to verify that $\bar{C} \vdash_\vartheta ex1^+$.

Fact e). $\bar{C} \not\vdash_\vartheta ex$ for any negative example ex. Assume this is not true, i.e. there exists a negative example ex and a substitution ϑ^- such that $\bar{C}\vartheta^- \subseteq ex$. Then, by definition of $\bar{C}$, we get $C\bar{\vartheta}\vartheta^- \subseteq ex$, a contradiction.

From facts a), b), c), d), and e), the clause $\bar{C}$ has necessarily the following form:

$$\bar{C}: \quad \leftarrow \bigwedge_{1 \leq i \leq t} p_i(Z_1^i, Z_2^i, Z_3^i) \\ \bigwedge_{1 \leq j \leq m} val_j(X'_j, X''_j)$$

where we have

(i) Z_j^i is the (X or Y) variable corresponding to the boolean variable appearing in the j-th position of the conjunct E_i (e.g., for $E_1 = x_1 \wedge \neg x_2 \wedge y_1$, we have $p_1(X_1, X_2, Y_1)$);

(ii) for each predicate val_j, either X'_j or X''_j must be the variable X_j, and the two variable are distinct.

Now, let T be the following truth value assignment for the X boolean variables of the formula Φ: $T(x_j) = true$ if $val_j(X_j, X''_j)$ appears in clause $\bar{C}$ (i.e., if X_j is the first argument of the predicate val_j); $T(x_j) = false$ if $val_j(X'_j, X_j)$ appears in clause $\bar{C}$ (i.e., if X_j is the second argument of the predicate val_j).

For assignment T, there exists no possible truth value assignment to variable in Y such that the 3DNF formula E is not satisfied. Indeed, if such an assignment existed, then $\bar{C}$ would satisfy the (negative) checking example, contradicting the hypothesis C is a solution of the ILP problem. □

It is worthwhile noting that now we have a precise characterisation of the sources of complexity we encounter in solving the above ILP problem: the subsumption test for checking whether a clause subsumes an example, and the choice of the positions of variables in the atoms. In fact, as we will see, even if we exactly know the format of the clause we are looking for, the complexity of the ILP consistency problem is Σ_2^P-complete. We next formalise the concept of format of a hypothesis, and the related ILP problems.

Definition 6. A *template* is a first order formula $\mathcal{T} \in \mathcal{L}_H$. The set of the instances of $\mathcal{T}$ is $\mathcal{I}(\mathcal{T}) = \{C \in \mathcal{L}_H \mid C = \mathcal{T}\vartheta \text{ for some substitution } \vartheta\}$

The *template* ILP-*learning* problem for an ILP setting $\mathcal{S}$ then defined as follows:

Definition 3.2 *Given* $\langle B, EX^+, EX^- \rangle$ *and a template* $\mathcal{T}$*, with* $B \in \mathcal{L}_B$*,* $EX^+ \in \mathcal{L}_E$*,* $EX^- \in \mathcal{L}_E$*, find an hypothesis* $H \in \mathcal{I}(\mathcal{T})$*, such that: (i)* $H \wedge B \vdash ex \quad \forall ex \in EX^+$*, and (ii)* $H \wedge B \not\vdash ex \quad \forall ex \in EX^-$.

The template *ILP*-consistency *problem for an ILP setting $\mathcal{S}$ is accordingly defined.*

In the *template* ILP-*consistency* problem for $\mathcal{S}_{HC}$, we know the template or "skeleton" of the solution we are looking for. The only information we do not get from the input is the identity/non-identity of variables in the desired instance of $\mathcal{I}(\mathcal{T})$.

However, the knowledge of the skeleton of the clause, does not reduce the complexity of the consistency problem for the ILP setting $\mathcal{S}_{HC}$. The following theorem is an easy consequence of the proof of Theorem 5.

Theorem 7. *The template ILP-consistency problem for $\mathcal{S}_{HC}$ is Σ_2^P-complete.*

4 Further Complexity Results

In this section, we focus on learning one 1,2-nondeterminate function-free Horn clause. First, we point out the complexity of ϑ-subsumption is still NP-complete, even under such a restriction. In fact, we further restrict this setting by allowing only one atom for each predicate in the clause.

Proposition 8. *Given a 1,2-nondeterminate function-free Horn clause C where all the atoms have distinct predicates and a ground Horn clause P, deciding whether $C \vdash_\vartheta P$ is* **NP***-complete. Hardness holds even if C and P are definite horn clauses.*

PROOF. Membership in **NP** is immediate, following the same reasoning of Lemma 4. Next we show the problem is **NP**-hard.

We reduce the **NP**-complete problem of deciding whether a graph G is 3-colorable to the problem of deciding whether a 1,2-nondeterminate function-free Horn clause C subsumes a ground Horn clause.

Let $G = (N, E)$ a graph and N and E be the sets of its nodes and edges, respectively. Let $|N| = n$ and $|E| = m$.

Consider the following 1,2-nondeterminate function-free Horn clause C:

$$h(B) \leftarrow \bigwedge_{e_i(x_j,x_k)\in E} pe_i(X_j, X_k) \bigwedge_{1\le j\le n} b_j(B, X_j)$$

Given an edge $e_i \in E$, define T_i to be the following conjunction:

$$pe_i(1,2) \wedge pe_i(1,3) \wedge pe_i(2,1) \wedge pe_i(2,3) \wedge pe_i(3,1) \wedge pe_i(3,2).$$

Then, P is the following Horn clause:

$$h(b) \leftarrow \bigwedge_{1\le i\le m} T_i \bigwedge_{1\le j\le n} b_j(b,1) \wedge b_j(b,2) \wedge b_j(b,3)$$

Now we claim that G is 3-colorable if and only if $C \vdash_\vartheta P$.

Only if part. Assume G is 3-colorable and let Col be a legal coloring for G, i.e. a mapping from the set N of nodes to the set $\{1,2,3\}$ of colors such that: (i) each node is colored with some color; and (ii) there exists no edge $e_i(x_j,x_k)$ such that $Col(x_j) = Col(x_k)$. Now, let ϑ be the following substitution for the variables appearing in C:

- $\vartheta(X_j) = Col(x_j)$ for $1 \leq j \leq n$;
- $\vartheta(B) = b$

It is clear that $C\vartheta \subseteq P$, since in each conjunction T_i, all (and only) the legal coloring for the endpoints of the corresponding edge are encoded.

If part. Suppose there exists a substitution ϑ such that $C\vartheta \subseteq P$. Let Col be the following coloring: $Col(x_j) = \vartheta(X_j)$ for $1 \leq j \leq n$. Note that each variable X_j must to be bound to a value in $\{1,2,3\}$ in order to subsume one of the atoms of the form $b_j(b,c)$, with $c \in \{1,2,3\}$.

Furthermore, if existed an edge $e_i(x_j,x_k)$ such that $Col(x_j) = Col(x_k) = c$, then the corresponding atom $pe_i(c,c)$ should belong to $C\vartheta$. Since, for any c, $pe_i(c,c)$ does not appear in T_i, we get $C\vartheta \not\subseteq P$, a contradiction. □

Let $\mathcal{S}_{1,2HC}$ be the setting $\langle \emptyset$, 1,2-nondeterminate function-free Horn clauses *whereallthe*
atoms have distinct predicates, $\vdash_\vartheta$, ground Horn clauses$\rangle$.

As a consequence of the above proposition, we know that for the setting $\mathcal{S}_{1,2HC}$, in general, given an hypothesis $H \in \mathcal{L}_H$ and an example $e \in \mathcal{L}_E$, the problem of deciding whether $e \in Cover(H)$ or not, is not feasible in polynomial time (unless **P** = **NP**). This evaluation hardness result, extends our knowledge of ILP settings which are not efficiently learnable. In particular, settings with arity of the predicates limited to 2 can be added to those considered by Cohen and Page in [6], where they showed hardness results for the setting $\mathcal{S}_{k,A}$, for any $A \geq 3$ (see Section 2.3 above).[2]

Moreover, since all the ILP settings at least as expressive as $\mathcal{S}_{1,2HC}$ inherit its evaluation hardness, we also extend a similar result stated in [18] only for the 1,2-nondeterminate function-free Horn clause. Indeed, Kietz and Džeroski showed that the consistency problem for the above setting (without requiring only one atom per predicate) is **NP**-hard.

In fact, we find out that the consistency problem for the setting $\mathcal{S}_{1,2HC}$ is located at the second level of the polynomial hierarchy.

Theorem 9. *The ILP-consistency problem for $\mathcal{S}_{1,2HC}$ is Σ_2^P-complete.*

PROOF. *Membership in Σ_2^P.* Let a set EX^+ of positive examples and EX^- of negative examples be given. Every appropriate clause C that subsumes the positive examples will contain at most $minPred(EX^+)$ predicates, where $minPred(EX^+)$ is the minimal number of predicates of the clauses in EX^+.

[2] Note that, in the proof of Proposition 8, C is not self resolving and P is not tautological, then the result holds for provability relation $\models$ too [15].

Therefore, we decide the ILP consistency problem at hand as follows. Guess a clause C with a number of predicates less than or equal to $minPred(EX^+)$ and perform the following checks: (i) for each clause P in EX^+, verify that $C \vdash_\vartheta P$; (ii) for each clause P' in EX^- verify that $C \nvdash_\vartheta P'$.

From Proposition 8, checking whether $C \vdash_\vartheta P$ can be done in **NP** (and checking that $C \nvdash_\vartheta P'$ can be done in co-**NP**). Therefore, by a polynomial number of calls to an **NP** oracle we are able to check that the guessed clause C is indeed a solution. The ILP-consistency problem for $\mathcal{S}_{1,2HC}$ is hence in Σ_2^P.

Σ_2^P-Hardness (Sketch). We transform deciding that a QBF $\Phi = \exists X \forall Y E$ is in $\text{QBF}_{2,\exists}$, where E is in 3DNF, into deciding whether there exists a 1,2-nondeterminate function-free Horn clause C such that $C \vdash_\vartheta EX^+$ and $C \nvdash_\vartheta ex$ $\forall ex \in EX^-$, where EX^+ and EX^- are given sets of ground Horn clauses (positive and negatives examples, respectively).

The proof is very similar to hardness part of the proof of Theorem 5, but we have to modify the p_i predicates in order to limit the arity to 2. In fact the two sources of complexity we have seen before does hold in this case as well. Namely, the evaluation hardness of the given setting (see Proposition 8) and the degree of freedom in the relative position of two variables. Note that, for the latter point, predicates of arity 2 are sufficient to encode the choice of either $val_j(X_j, X'_j)$ or $val_j(X'_j, X_j)$. □

References

1. L.M. Adleman. Molecular Computation of Solutions to Combinatorial Problems. *Science* 266, pp. 1021–1024, 1994.
2. L. D. Baxter. The NP-completeness of Subsumption. Unpublished manuscript, 1977.
3. P. Cholewinski, V.W. Marek, and M. Truszczynski Default Reasoning System DeReS *Proc. Fifth Intl. Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, Cambridge, MA, Nov.5-8, 1996.
4. W. Cohen. PAC-Learning Recursive Logic Programs: Efficient Algorithms. *Journal of Artificial Intelligence Research*, 2: 501–539, 1995.
5. W. Cohen. PAC-Learning Recursive Logic Programs: Negative Results. *Journal of Artificial Intelligence Research*, 2: 541–573, 1995.
6. W. Cohen and C. D. Page. Polynomial Learnability and Inductive Logic Programming - Methods and Results. *New Generation Computing*, 13(3-4): 369–409, 1995.
7. L. De Raedt and S. Džeroski. First order *jk*-clausal theories are PAC-learnable. *Artificial Intelligence*, 70: 375–392, 1994.
8. J. Dix and M. Müller. Implementing Semantics of Disjunctive Logic programs Using Fringes and Abstract properties. *Proc. Second Intl. workshop on Logic Programming and Nonmonotonic reasoning (LPNMR-93)*, Lisbon, Portugal, July 1993, pp. 43–59, MIT Press.
9. T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 15(3/4):289–323, 1995.

10. T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Nonmonotonic Reasoning. In *Proc. 4th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '97)*, Lecture Notes in AI (LNAI), J. Dix, U. Furbach, and A. Nerode Eds., Springer, Berlin, 1997 (to appear).
11. T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Trans. on Database Syst.*, September 1997. To appear.
12. M. R. Garey and D. S. Johnson. *Computers and Intractability – A guide to the Theory of NP-completeness.* Freman, San Francisco, CA, 1979.
13. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Logic Programming: Proc. Fifth Intl Conference and Symposium*, pp. 1070–1080, Cambridge, Mass., 1988. MIT Press.
14. E. M. Gold. Language Identification in the Limit. *Information and Control*, 10:447–474, 1967.
15. G. Gottlob. Subsumption and Implication. *Information Processing Letters*, 24:109–111, 1987.
16. G. Gottlob. Complexity Results for Nonmonotonic Logics. *J. Logic and Computation*, 2(3):397–425, June 1992.
17. D. S. Johnson. A Catalog of Complexity Classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Publishers B.V. (North-Holland), 1990.
18. J.U. Kietz and S. Džeroski. Inductive logic programming and learnability. *SIGART Bulletin* 5(1): 22–32 (Special issue on Inductive Logic Programming), 1994.
19. D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's model. *Artificial Intelligence*, 36(2):177–221, 1988.
20. R.J. Lipton. Using DNA to solve NP-complete Problems. *Priceton University.*
21. W. Marek and M. Truszczyński. Autoepistemic Logic. *JACM*, 38(3):588–619, 1991.
22. S. H. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.
23. *Inductive Logic Programming*, S. H. Muggleton ed., Academic Press, London, 1992.
24. S. H. Muggleton and L. De Raedt. Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming*, 19,20:629–679, 1994.
25. Muggleton, S., and Page, D., A learnability model for universal representations. In *Proc. Fourth International Workshop on Inductive Logic Programming*, pages 139-160. GMD, Bonn, 1994.
26. C. D. Page and A. M. Frish. Generalization and Learnability: a study of constrained atoms. In *Inductive Logic Programming*, pp.29–61, S. H. Muggleton ed., Academic Press, London, 1992.
27. G. D. Plotkin. A note on Inductive Generalization. In *Machine Intelligence*, pp. 153–163, B. Meltzer and D. Michie eds., American Elsevier, 1970.
28. R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.
29. J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.
30. R. E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5:197–227, 1990.
31. Diana Rooß and Klaus Wagner. On the Power of DNA Computation. *Information and Computation*, 131(2):95–109, 1996.
32. L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27:1134–1142.