On the Complexity of Inductively Learning Guarded Clauses

Andrei Draghici, Georg Gottlob, Matthias Lanzinger
University of Oxford
andrei.draghici@stcatz.ox.ac.uk, georg.gottlob@cs.ox.ac.uk, matthias.lanzinger@cs.ox.ac.uk

Abstract

We investigate the computational complexity of mining guarded clauses from clausal datasets through the framework of inductive logic programming (ILP). We show that learning guarded clauses is NP-complete and thus one step below the Σ_2^P-complete task of learning Horn clauses on the polynomial hierarchy. Motivated by practical applications on large datasets we identify a natural tractable fragment of the problem. Finally, we also generalise all of our results to k-guarded clauses for constant k.

Introduction

Finding complex relationships in large datasets with current data analysis methods primarily relies on the computation of statistical measures or simple aggregations over numerical data. Especially when relationships between some parts of the data are conditional on other data elements, standard methods such as correlation analysis reach their limits. To better identify conditional relationships, one can alternatively approach the problem by trying to learn logical relationships in the data, i.e., searching for non-trivial logical formulas that hold true over the data set.

Learned logical formulas should ideally be intuitively interpretable as well as usable in automated reasoning. A natural fit for these requirements are logical rules (i.e., Horn clauses) as used in Datalog (Ullman 1989) and the associated task of finding rules that are satisfied in a dataset is often called *rule mining* (Fürnkranz, Gamberger, and Lavrac 2012). Not only are modern rule-based languages highly expressive and theoretically well understood, but there also exist real-world systems that are capable of efficient reasoning over large amounts of data, e.g. (Bellomarini, Sallinger, and Gottlob 2018). In combination with such reasoning systems, learned rules can also be used for data synthesis, anomaly detection, and similar tasks.

This paper investigates the underlying theoretical problems when viewing the rule mining problem through the lens of inductive logic programming (ILP). Recall, in the ILP setting we are given a set of positive examples and a set of negative examples and wish to find some logical formula (the hypothesis) that subsumes all of the positive examples while not subsuming any of the negative examples. We are particularly interested in the computational complexity of the problem and therefore consider the associated decision problem, also referred to as the *ILP consistency problem*, whether any solution exists for the given sets of examples.

The complexity of the ILP consistency problem depends critically on the logical languages that are used for examples and the hypothesis. In line with our motivation of learning rules, particular historical attention has been given to the case where examples and hypotheses are Horn clauses, for which the ILP consistency problem was shown to be Σ_2^P-complete (Gottlob, Leone, and Scarcello 1997). While some problems in NP can now be solved efficiently in practical settings, the second level of the polynomial hierarchy is still generally out of reach for even moderately sized problems. In the context of mining rules from large datasets, Σ_2^P-completeness is therefore clearly prohibitive and limits potential applications to only small and simple scenarios.

To address this challenge we consider the learning of *guarded clauses*, i.e., we restrict the language of the hypothesis to only guarded clauses. Informally, a clause is said to be *guarded* if there exists one literal in the clause that *covers* all variables that occur in the clause. Our interest in guarded clauses is motivated from two sides. First, as discussed below, we see that learning guarded clauses can reduce the complexity of the problem. Second, in common rule-based languages, e.g., Datalog or Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2009), programs consisting only of guarded rules also enjoy various desirable computational properties (Gottlob, Grädel, and Veith 2002; Calì, Gottlob, and Lukasiewicz 2009)[1]. While guarded rules are of particular interest in practice, we present our results for the more general problem of learning guarded clauses, i.e., disjunctive rules. However, all results hold also for Horn clauses.

Importantly, guarded clauses and corresponding rule-based languages are still highly expressive. The guarded fragment of Datalog$^{\pm}$ captures popular logical languages, such as the description logics DL-Lite$_{\mathcal{R}}$ and $\mathcal{EL}$ which form the theoretical foundation of the OWL 2 QL and EL profiles,

[1]Guarded rules in the context of Datalog$^{\pm}$ and guarded clauses have slight technical differences. These differences are inconsequential to our results which all also apply to learning guarded rules in way the term is used in Datalog$^{\pm}$, see the discussion after Algorithm 1.

respectively (Calì, Gottlob, and Lukasiewicz 2012). In addition to the expressiveness, reasoning over guarded clauses can be more efficient than over general clauses (Calì, Gottlob, and Lukasiewicz 2012). The following example aims to further clarify our setting.

Example 1. Consider the following clauses, which make up a set of positive examples $\{E_1^+, E_2^+\}$ and the single negative example E_1^- (with constants a, b, c, d, e).

$$E_1^+ : \neg\mathsf{TalkAbout}(a,b,a) \vee \neg\mathsf{FanOf}(a,a) \vee \neg\mathsf{Influences}(a,b) \vee \mathsf{FanOf}(b,a),$$

$$E_2^+ : \neg\mathsf{TalkAbout}(a,c,d) \vee \neg\mathsf{FanOf}(a,d) \vee \neg\mathsf{Influences}(a,c) \vee \mathsf{FanOf}(c,d) \vee \mathsf{Parent}(c,b)$$

$$E_1^- : \neg\mathsf{TalkAbout}(d,b,e) \vee \neg\mathsf{Influence}(d,b) \vee \mathsf{FanOf}(d,e)$$

The following guarded clause (written as a rule) is a natural ILP-hypothesis that can be learned from the above examples.

$$\mathsf{FanOf}(y,z) \leftarrow \mathsf{TalkAbout}(x,y,z), \mathsf{FanOf}(x,z), \mathsf{Influences}(x,y)$$

Intuitively, the rule can be interpreted as: if x and y talk about z, x is a fan of z, and x influences y, then y will also be a fan of z. Here, the atom $\mathsf{TalkAbout}(x,y,z)$ contains all three variables that are used in the clause, and therefore *guards* the clause. Alternatively, if we drop the guardedness requirement, the following clause would also be a valid hypothesis

$$\neg\mathsf{Influences}(x,y) \vee \mathsf{FanOf}(y,z)$$

Note that it is difficult to interpret this clause intuitively, especially when viewed at as a rule: if x influences y, then y is a fan of t.

Related Work Inductive logic programming and learning clauses in particular is a classic theme of AI research and efficient algorithms for learning clauses have been the topic of a number of classic works, e.g. (Muggleton 1991; Muggleton and Feng 1990; Kietz and Lübbe 1994). The complexity of learning clauses has been studied in many other contexts and under a broad range syntactical restrictions. One particularly important case is the learning of Horn clauses. There, the ILP consistency problem is Σ_2^P-hard (Gottlob, Leone, and Scarcello 1997), even without the presence of background knowledge. Despite the large body of work on learning clauses, to the best of our knowledge, guardedness has not yet been studied in this context. One common feature of ILP settings that we do not explicitly consider here is specifying target predicates (cf., (Cohen and Jr. 1995)). Our algorithm and hardness proof can be adapted to cases where a specific target predicate is considered. Note that specifically for our algorithm, even enumerating a set of all *representative* solutions is tractable and choice of a certain target predicate is therefore not a critical factor.

Recently, the inductive synthesis of Datalog programs, using a variety of different techniques, has received particular attention (Si et al. 2019; Raghothaman et al. 2020). The resulting systems are able to synthesise complex logic programs, including advanced use of recursion, background knowledge and predicate invention.

However, our motivation of rule mining in large datasets is not fully aligned with this direction of research. In our setting we prefer learning many individual rules instead of complex programs while also requiring stronger upper bounds on the computational complexity to deal with large inputs. Our results may be applicable to the synthesis of *guarded* Datalog programs when considered in conjunction with some of the methods noted above. For further details on new developments in the synthesis of logic programs and various other directions of current ILP research, we defer to a recent survey article (Cropper, Dumancic, and Muggleton 2020).

Our Contributions Our goal is to study the computational complexity of learning guarded rules and find expressive but tractable fragments that can support real-world rule mining. We show that in the general case, the problem is no longer Σ_2^P-hard but still intractable. However, we are able to identify the source of the intractability and demonstrate that an important fragment of the problem is indeed tractable. The full contributions of this paper can be summarised as follows:

1. First we show that guardedness in general improves the complexity in versus the restriction to Horn clauses. In particular, the problem moves one step down on the polynomial hierarchy from Σ_2^P to NP.
2. We show not only NP-completeness of the general problem but also for various common further restrictions such as bounded arity of relations and bounded number of variables in the hypothesis.
3. In response, we identify the notion of straight clauses. We show that learning guarded clauses from a clausal dataset is tractable if the positive examples are straight. We present a polynomial time algorithm and furthermore, show that our algorithm also allows tractable enumerate of an class of representative solutions of the problem.
4. Finally, we argue that our results also apply analogously to the much more general case of guardedness by not one literal, but a constant number of literals (so-called k-guardedness).

Preliminaries

We assume that the reader is familiar with typical notions of computational complexity as defined in (Papadimitriou 2007), in particular polynomial time reductions and the first two levels of the polynomial hierarchy.

We begin by recalling some important standard notions of inductive logic programming. We roughly follow the presentation from (Kietz and Dzeroski 1994). We assume basic familiarity with standard terminology of first-order logic as used, e.g., in (Fitting 1996). In general, for a literal L we will use $\mathrm{vars}(L)$ to denote the set of variables occurring in the atom. Likewise, we write $\mathrm{terms}(L)$ for the set of all terms occurring in L, and $\mathrm{ar}(L)$ for the arity of L.

Let $\mathcal{L}$ be the language of first order logic. An *ILP setting* is a tuple $\langle \mathcal{L}_B, \mathcal{L}_H, \vdash, \mathcal{L}_E \rangle$ where $\mathcal{L}_B, \mathcal{L}_H, \mathcal{L}_E$ are subsets of $\mathcal{L}$ and $\vdash$ is a correct provability relation.

For a set of formulas $E, T \vdash E$ is short for $T \vdash e$ for every $e \in E$, and $T \not\vdash E$ is short for $T \not\vdash e$ for every $e \in E$. Hence, a valid hypothesis must subsume all positive examples and none of the negative examples.

The *ILP consistency problem* for a ILP setting $\mathcal{S}$ denotes the following decision problem: Given sets $B, Ex^+ \cup Ex^-$ where $B \subseteq \mathcal{L}_B$, and $Ex^+, Ex^- \subseteq \mathcal{L}_E$, does there exist a *hypothesis* $H \in \mathcal{L}_H$ such that $B \cup \{H\} \vdash Ex^+$, $B \cup \{H\} \not\vdash Ex^-$, and $B \cup \{H\} \not\vdash \bot$ ($B \cup H$ is consistent). In such an instance, B is commonly referred to as the *background knowledge*, while Ex^+, Ex^- are the positive and negative *examples*, respectively. Furthermore, we will refer to any such hypothesis as a *solution* to the respective instance. By slight abuse of notation we will also consider ILP settings where the languages for the examples $\mathcal{L}_E$ is split into separate languages for the positive and the negative examples. Thus Ex^+ above will consist of clauses from a language $\mathcal{L}_{E^+}$, while Ex^- are clauses from a possibly different language $\mathcal{L}_{E^-}$. The languages can of course still overlap and this variation of ILP settings has no effect on other definitions.

As is common in the field, we will consider specifically languages of first order clauses. It will be convenient to treat a clause as a set of the individual disjuncts and we will do so implicitly throughout this paper. In this spirit we also assume no duplicate literals in clauses. It will be useful to distinguish between a relation name occurring in a positive and a negative literal. We use the term *signed relation name* to refer to a relation name together with the polarity of the literal in which it occurs. Following common practice in the field, we focus on *θ-subsumption* $\vdash_\theta$ as a provability relation here. Recall, we say that a clause ϕ θ-subsumes a clause ψ, i.e., $\phi \vdash_\theta \psi$, if there exists a variable substitution θ such that $\phi\theta \subseteq \psi$. As we will always use $\vdash_\theta$ as the provability relation, we will generally drop the subscript and use $\vdash$ to implicitly represent θ-subsumption.

We are now ready to recall the important result for Horn clauses mentioned in the introduction. Let $\mathcal{S}_{HC}$ be the ILP setting $\langle \emptyset, \text{function-free Horn clauses}, \vdash_\theta, \text{ground clauses} \rangle$.

Proposition 1 ((Gottlob, Leone, and Scarcello 1997)). *The ILP consistency problem for $\mathcal{S}_{HC}$ is Σ_2^P-hard. When the learned hypothesis is assumed to be polynomially bounded in the size of the examples, the problem is Σ_2^P-complete.*

For the rest of this paper we are interested in the problem of learning *guarded* clauses. We say that a clause C is *guarded* if there is some literal $G \in C$ such that $\bigcup_{R \in C} \text{vars}(R) \subseteq \text{vars}(G)$. Guardedness can also be generalised to so-called *k-guarded* clauses, where some group of k literals in the clause covers all variables of the clause. Much of this paper will focus on the ILP setting $\mathcal{G} = \langle \emptyset, \text{function free guarded clauses}, \vdash_\theta, \text{ground clauses} \rangle$. Since $\mathcal{G}$ does not allow for background knowledge, we will treat instances for the ILP consistency problem of $\mathcal{G}$ simply as tuples of positive and negative examples.

NP-Completeness of Guarded ILP Consistency

Note that the language of guarded clauses is incomparable to the language of Horn clauses. Our setting is therefore not simply a more restricted version of previously studied work but an alternative restriction, which we will show to have more desirable computational properties.

First, in this section, we show that the ILP consistency problem for $\mathcal{G}$ is NP-complete. This represents a significant improvement over the complexity of $\mathcal{S}_{HC}$. The proof details will allow us to identify a practically useful tractable fragment of the problem, which we present in next section.

We will first show a reduction from the hitting string problem (Garey and Johnson 1979) to establish NP-hardness. Afterwards we present an argument for NP membership. Note that membership of the problem is not straightforward. It is not trivial whether it is always sufficient to guess a polynomially bounded hypothesis. Furthermore, it is not easy to verify a guess in polynomial time since checking θ-subsumption is NP-hard in general (Kietz and Lübbe 1994; Garey and Johnson 1979).

Reduction from Hitting String

In the *hitting string* problem we are given a finite set of strings $S = \{\mathbf{s_1}, \dots, \mathbf{s_m}\}$, all of length n, over the alphabet $\{0, 1, *\}$. The task is to decide whether there exists a *binary* (over the alphabet $\{0, 1\}$) string $\mathbf{x} = x_1 x_2 \dots x_n$ such that for each $\mathbf{s_i} \in S$, $\mathbf{s_i}$ and $\mathbf{x}$ agree in at least one position. Such a string $\mathbf{x}$ is called a *hitting string* for the set S. The hitting string problem is known to be NP-complete (Garey and Johnson 1979).

Theorem 2. *The ILP consistency problem for $\mathcal{G}$ is* NP*-hard. The problem remains* NP*-hard even when hypothesis and example languages are additionally restricted to clauses with any combination of the following properties:*

- *Horn,*
- *arity at most 2,*
- *at most 2 variables in the hypothesis,*
- *and no repetition of terms in a single literal.*

Proof. Proof is by reduction from the hitting string problem. Let $S = \{\mathbf{s_1}, \dots, \mathbf{s_m}\}$ be set of length n strings over the alphabet $\{0, 1, *\}$. Let the positive examples Ex^+ be the set $\{C_i \mid 0 \leq i \leq n\}$ with the respective clauses defined as

$$\begin{aligned} C_0 = \quad & G(a,b) \vee \textstyle\bigvee_{1 \leq j \leq n} (A_j(a) \vee B_j(b)) \\ C_i = \quad & G(a,b) \vee G(b,a) \vee A_i(a) \vee B_i(a) \vee \\ & \textstyle\bigvee_{1 \leq j \leq n, j \neq i} (A_j(a) \vee A_j(b) \vee B_j(a) \vee B_j(b)) \end{aligned}$$

For string $\mathbf{s}_i \in S$, we write $s_{i,j}$ to refer to the symbol at position j of the string. Let set of negative examples Ex^- be the set $\{D_i \mid 0 \leq i \leq n\} \cup \{N_i \mid 0 \leq i \leq m\}$ where the individual clauses are as follows

$$\begin{aligned} D_0 = \quad & \textstyle\bigvee_{1 \leq j \leq n} (A_j(a) \vee A_j(b) \vee B_j(a) \vee B_j(b)) \\ D_i = \quad & G(a,b) \vee \textstyle\bigvee_{1 \leq j \leq n, j \neq i} (A_j(a) \vee B_j(b)) \\ N_i = \quad & G(a,b) \vee \textstyle\bigvee_{j : s_{i,j} = 1} B_j(b) \vee \bigvee_{j : s_{i,j} = 0} A_j(a) \\ & \vee \textstyle\bigvee_{j : s_{i,j} = *} (A_j(a) \vee B_j(b)) \end{aligned}$$

Now suppose there is a function-free guarded clause H such that $H \vdash Ex^+$ and $H \vdash Ex^-$. We first make some observations on the structure of H and then argue how this connects to the original hitting string instance. First, observe that any hypothesis H must contain some atom for relation G to not subsume D_0. On the other hand, by C_0, we see that both positions must contain different variables and only one of $G(x,y)$ and $G(y,x)$ is in H. Hence, w.l.o.g., we assume only $G(x,y)$ is in H.

Since H must subsume C_0, there can be no atoms $A_i(y)$ or $B_i(x)$ in H. In consequence, since D_i may not be subsumed, at least one of $A_i(x)$ or $B_i(y)$ must occur in H. Indeed, because H needs to subsume C_i it follows that only exactly one of $A_i(x)$ or $B_i(y)$ is in H for every $1 \leq i \leq n$. Hence, we know that H is always of the form

$$G(x,y) \vee \bigvee_{1 \leq i \leq n} \Gamma_i, \quad \text{where } \Gamma_i \text{ is either } A_i(x) \text{ or } B_i(y)$$

Note that H is clearly always guarded. We can associate a binary string $\mathbf{x}(H)$ of length n to each clause H of this form by simply setting position i of the string to be 1 if $A_i(x) \in H$, or 0 if $B_i(y) \in H$.

We are now ready to show the correctness of the reduction, in particular we argue that $\mathbf{x}(H)$ is a hitting string for S if and only if $H \vdash Ex^+$ and $H \not\vdash Ex^-$.

Assume $\mathbf{x}$ is a hitting string for S and let H be the clause associated to $\mathbf{x}$. From our arguments above we know that the only thing left to check is whether H does not subsume any of the N clauses. Every string $\mathbf{s}_i \in S$ agrees with $\mathbf{x}$ on some position, i.e., $\mathbf{s}_{i,j} = \mathbf{x}_j$. If such a $\mathbf{s}_{i,j}$ is 1, then $A_j(x) \in H$ but $A_j(a) \notin N_i$. The analogous reasoning holds for the case where the position contains 0. Since $\mathbf{x}$ is a binary string, these are the only two options and we see that for every string $\mathbf{s}_i \in S$, the clause N_i is not subsumed by H. Hence, $H \vdash Ex^+$ and $H \not\vdash Ex^-$.

For the other side of the implication, suppose there is a guarded clause H with $H \vdash Ex^+$, $H \not\vdash Ex^-$. From our arguments before we know the structure of H and that the clause determines a unique binary string $\mathbf{x}(H)$. Now, for every $0 \leq i \leq m$ we have that $H \not\vdash N_i$. By inspection of H and N_i it follows that for some $0 \leq j \leq n$, either $A_j(x) \in H$ and $\mathbf{s}_{i,j} = 1$, or $B_j(y) \in H$ and $\mathbf{s}_{i,j} = 0$. In either case, by definition of $\mathbf{x}(H)$ we see that the j-th position of $\mathbf{x}(H)$ agrees with the value of $\mathbf{s}_{i,j}$. Since this holds for all positions, it follows that $\mathbf{x}(H)$ agrees with every string in S at some position, i.e., $\mathbf{x}(H)$ is a hitting string for S.

This establishes the NP-hardness for $\mathcal{G}$. By inspection of the clauses in the reduction, we can deduce that the problem remains hard under the stated additional restrictions. All of the clauses are positive, i.e., polarity plays no role in the argument. It is also easy to adapt each clause to be Horn instead (make all atoms in the above reduction negative, add a new redundant literal $P(a)$ to every clause). Lastly, only arity 2 atoms, 2 variables in H and 2 constants in the examples are necessary. □

Note that our reduction also demonstrates that the restriction to guardedness is not itself a source of difficulty. The above argument does not require the restriction to guarded hypothesis. It follows from our observations on the form of solutions to our reduction that any clause that subsumes all of Ex^+ and none of Ex^- must be guarded.

NP Membership

We first establish that it is sufficient to guess a polynomially bounded hypothesis. Note that this is not true in every setting and previous work has instead regularly considered the *bounded ILP consistency problem* (Alphonse and Osmani 2009; Gottlob, Leone, and Scarcello 1997), where explicit constraints on the hypothesis size are considered as part of the decision problem.

While guardedness still allows for exponentially large solutions, most literals in such a solution will be redundant. In particular, we can observe that if a guarded hypothesis does not subsume a clause, then there are only a few concrete variable assignments for which subsumption must fail in addition to those cases where the guard itself can not be subsumed. A solution then needs only one literal per such case to guarantee non-subsumption of the clause. We can therefore observe the following bound (a full argument is given in the technical appendix).

Lemma 3. *Assume the instance Ex^+, Ex^- for the ILP consistency problem for $\mathcal{G}$ has a solution. Then it also has a solution H with $|H| \leq 1 + \sum_{E \in Ex^-} |E|$.*

The natural guess and check procedure for the ILP consistency problem is to guess a hypothesis and then check (non-)subsumption for each of the examples. However, in general this does not result in an NP algorithm since deciding whether $\phi \vdash \psi$ is NP-complete even for function-free Horn clauses (Kietz and Lübbe 1994).

In the guarded setting, subsumption can be decided more efficiently. It is not difficult to prove directly that subsumption by a function-free guarded Horn clause is tractable. Due to our later interest in k-guardedness it will be more convenient to adapt a more general result on the complexity of finding homomorphisms.

It is straightforward to translate the problem $\phi \vdash \psi$, where ϕ is a function-free clause, to a the question of whether there exists a homomorphism from a relational structure representing ϕ to a structure representing ψ. The computational complexity of homomorphism checking is well understood (Grohe 2007; Gottlob, Leone, and Scarcello 2002), and by applying this knowledge to our setting we can make the following important observation.

Lemma 4. *Let k be an integer constant, let ϕ and ψ be clauses such that ϕ is k-guarded. Then it is tractable to decide whether $\phi \vdash_\theta \psi$.*

Naturally, the complementary problem $\phi \not\vdash \psi$ is tractable under the same circumstances. Since we know from Lemma 3, that it is sufficient to guess a polynomially bounded H, it is clear that the ILP consistency problem for $\mathcal{G}$ is in NP. In combination with Theorem 2 we arrive at the main result of this section.

Theorem 5. *The ILP consistency problem for $\mathcal{G}$ is* NP-*complete.*

Tractability for Straight Positive Examples

In the previous section we have seen that considering guarded clauses instead of Horn clauses improves the complexity of the consistency problem form Σ_2^P-completeness to NP-completeness. However, in the context of our original motivation of rule mining on large datasets this may still be infeasible. Furthermore, Theorem 2 illustrates that even severe restrictions to the clause languages are insufficient to achieve a polynomial time algorithm (assuming NP $\neq$ PTIME).

In this section we introduce the notion of straight clauses which will allow us to define a tractable fragment of the problem that can still express many common notions, such as recursive rules. We then prove some important theoretical properties under this new restriction and ultimately give a polynomial time algorithm. Finally, we briefly discuss how key results and our algorithm generalise to k-guardedness.

Definition 1 (Straight Clauses)**.** A clause ϕ is called *straight* if no relation name occurs twice in ϕ with the same polarity.

Note that straight clauses can still express recursive rules since the same relation can appear twice with differing polarity. For example, all clauses and the resulting hypothesis in Example 1 are straight clauses. In the context of Datalog rules, straight rules correspond to rules with no self-joins.

It will not be necessary to restrict all clauses to be straight. Indeed, we will only require positive examples to be straight clauses, whereas the negative clauses and the hypothesis will not need to be straight. To that end we define the new ILP setting $\mathcal{G}^*$ as the setting $\mathcal{G}$ where the positive examples additionally have to be straight clauses.

Guarded Hypotheses for Straight Clauses

We first introduce some new notation. In a clause C where a signed relational name P occurs only once we will use $\pi_{P,i}^C$ to denote the term at the i-th position of the literal over P.

Definition 2 (Relative Shields)**.** Let Q and P be relational names occurring in a straight clause C. The *relative shields* of position i of Q are defined as the set

$$S_{P,Q,i}^C = \{j \mid \pi_{Q,j}^C = \pi_{P,j}^C\}$$

That is, the relative shields are all the positions of P that contain the exact same term as position i of Q.

Let E be a set of straight clauses that all contain P and Q. The relative shield of position i of Q by P (w.r.t. E) is the set

$$S_{P,Q,i}^E = \bigcap_{C \in E} S_{P,Q,i}^C$$

Example 2. Consider the following two clauses

$$P(a,a,b,a),\ Q(a)$$
$$P(a,b,a,b),\ Q(b)$$

The relative shield of position 1 of Q by P is $\{1,2,4\}$ in the first clause and $\{2,4\}$ in the second clause. Thus, for the set of clauses we have the relative shield $\{2,4\}$. In situations where P is the guard of the hypothesis H the relative shield corresponds to the variables from the guard clause in H that can be used in the respective position of Q. That is, if $P(x,y,x,z)$ is the guard in H, then only $Q(y)$ or $Q(z)$ are possible candidates for the hypothesis.

The following lemma formalises the observation made at the end of the example. It shows how the relative shield and variable choice are tightly connected in guarded clauses.

Lemma 6. *Let E be a set of straight clauses and let H be a clause with guard P such that $H \vdash E$. For every literal $Q(x_1,\dots,x_n) \in H$, we have that for every $1 \le i \le \mathrm{ar}(Q)$, if $x_i = \pi_{P,j}^H$, then $j \in S_{P,Q,i}^E$.*

We will see that beyond Lemma 6, the relative shield will play a special role if we consider a particular guard, namely the least general one.

Definition 3 (Least General Induced Guard)**.** Let E be a set of straight ground clauses that all contain the signed relation name P. The least general induced guard (lgig) for P (induced by E) is the literal $P(\bar{x})$ with the least number of distinct variables such that $P(\bar{x}) \vdash \{P(\bar{a})\}$, for all $P(\bar{a})$ in any clause in E.

It is easy to observe that the lgig for P has the same variables exactly in those positions that contain the same term in every clause. That is, $x_i = x_j$ if and only if $\forall C \in E\colon \pi_{P,i}^C = \pi_{P,j}^C$, where x_k refers to the variable in position k of the lgig. It follows that the lgig is unique up to isomorphism.

In the next step, we will show how using an lgig as guard can determine the exact argument list of all literals in the hypothesis. This is an important observation for our eventual algorithm. Furthermore, it will allow us to show that it is always sufficient to consider only lgigs as guards in our scenario.

Lemma 7. *Let E be a set of straight ground clauses and let H be a clause guarded by the lgig for some signed relation name P such that $H \vdash E$. For every literal $Q(\dots) \in H$, we have that for every $i \in ar(Q)$ and every $j,k \in S_{P,Q,i}^E$ we have $\pi_{P,j}^H = \pi_{P,k}^H$.*

Proof. Suppose the statement fails for $Q(x_1,\dots,x_n) \in H$ at position i. That is, there are $j,k \in S_{P,Q,i}^E$ such that there are different variables at positions j and k in the guard P, i.e., $\pi_{P,j}^H \neq \pi_{P,k}^H$. Observe that if j,k are both relative shields, then it holds that $\pi_{P,j}^C = \pi_{Q,i}^C = \pi_{P,k}^C$ for all $C \in E$. Therefore, for every $C \in E$ there exists a θ such that $H\theta \subseteq C$ where θ maps $\pi_{P,j}^C$ and $\pi_{P,k}^C$ to the same term. Thus, let θ' be the substitution that maps $\pi_{P,j}^C \mapsto \pi_{P,k}^C$ and everything else to identity. Clearly, $H\theta' \vdash E$ and the guard in $H\theta'$ has symbol P and fewer distinct variables than the guard in H, hence contradicting the assumption of the guard in H being the lgig. $\square$

Since the above lemma uniquely determines the argument list of each literal occurring in an lgig guarded hypothesis, it also follows that no signed relation name can occur twice in the hypothesis. Hence, the hypothesis is also always straight.

Another important property of the lgig is related to the following notion of specialisation. We call a variable substitution θ a *specialisation* if it maps from the variables X to a subset $X' \subset X$ such that $\theta(x) = x$ for all $x \in X'$. The following lemma establishes not only that if there is a solution, there is also a solution guarded by the lgig, but that any guard in a solution can be specialised to be an lgig.

Lemma 8. *Let E be a set of straight grounded clauses and let H be a clause with guard literal $P(\bar{x})$ such that $H \vdash E$. There exists a specialisation θ such that $P(\bar{x})\theta$ is the lgig of P induced by E.*

Proof. Let $P(\bar{y})$ be the lgig of P induced by E. Recall that $y_i = y_j$ if and only if $\forall C \in E \colon \pi^C_{P,i} = \pi^C_{P,j}$. Observe that if two positions i, j of P differ in some clause in E (and if $H \vdash E$), then the variables at the same positions of $P(\bar{x})$ in H can not be the same. Otherwise the same variable would have to map to different terms to match the single occurrence of P in the clause.

With this established, we see that in $P(\bar{x})$ we have $x_i = x_j$ only if $y_i = y_j$ in the lgig $P(\bar{y})$. Thus, clearly the substitution $\theta : x_i \mapsto y_i$ for all $1 \leq 1 \leq \mathrm{ar}(P)$ is a specialisation and $P(\bar{x})\theta$ is the lgig of P induced by E. $\square$

From these key lemmas we can derive the following key theorem for the setting $\mathcal{G}^*$. A full proof of the missing details is given in the technical appendix.

Theorem 9. *Let Ex^+ be a set of straight ground clauses, and let Ex^- be a set of ground clauses. If there exists a guarded clause H such that $H \vdash Ex^+$ and $H \not\vdash Ex^-$, then there also exists a guarded clause H' with the following properties:*

1. *The guard of H' is a least general induced guard (induced by Ex^+).*
2. *H' is a straight clause.*
3. *$H' \vdash Ex^+$ and $H' \not\vdash Ex^-$.*

A Polynomial-Time Algorithm

The results of the previous section show that we can use least general induced guards to fix a mapping of terms in a positive example to variables in the hypothesis. By Lemma 7, this mapping, in combination with a ground literal from the example, determines the only possible way to include a corresponding literal (with the same polarity and relation symbol) in the hypothesis.

Lemma 7 is a crucial observation for our algorithm. The algorithm iteratively adds literals from a positive example to the hypothesis by replacing the terms there with variables. In general, the choice of how exactly terms are replaced by variables leads to combinatorial explosion. However, Lemma 7 states that we can limit our search to one particular choice if the hypothesis is guarded by an lgig.

Definition 4. Let E be a set of straight ground clauses and let H be a clause guarded by the lgig for P. Let L be a literal in some clause of E. The *lgig map* μ of L is a function $\mathrm{terms}(L) \to \mathrm{vars}(H)$ such that $\mu(t_i) = \pi^H_{P,j}$ where $j \in S^E_{P,L,i}$ and t_i is the term at position i of L.

It follows from Lemma 7 that such a map always exists – and is uniquely determined – if a literal with the same name and polarity as L is part of a solution guarded by the lgig for P. Note that no lgig map exists if the relative shield is empty at any position i.

Algorithm 1: A polynomial time algorithm for ILP consistency for setting $\mathcal{G}^*$.

```
   input  : Examples $Ex^+ \neq \emptyset$, $Ex^-$ as sets of
            grounded straight clauses.
   output: Guarded clause $H$ s.t. $H \vdash Ex^+$ and
            $H \not\vdash Ex^-$
1  $C \leftarrow$ an arbitrary clause in $Ex^+$;
2  for $G \in C$ do
3      $G' \leftarrow$ be the least general induced guard for $G$
         induced by $Ex^+$;
4      $H \leftarrow \{G'\}$;
5      if $H \not\vdash Ex^-$ then
6          return $H$;
7      for $L \in C \setminus \{G\}$ do
8          if an lgig map of $L$ w.r.t. $G'$ exists then
9              $\mu \leftarrow$ lgig map of $L$ w.r.t. lgig $G'$;
10             if $H \cup \{\mu(L)\} \vdash Ex^+$ then
11                 $H \leftarrow H \cup \{\mu(L)\}$;
12     if $H \not\vdash Ex^-$ then
13         return $H$;
14 Reject
```

With the above definition, we are now ready to present Algorithm 1, our polynomial time algorithm for the ILP consistency problem for $\mathcal{G}^*$. To avoid redundant checks throughout the algorithm we assume, w.l.o.g., that the instance is non-trivial, i.e., that $Ex^+ \neq \emptyset$ and that all signed relation names occur in all clauses of Ex^+. Clearly, any signed relation name that does not occur in all clauses can never be part of the hypothesis, as it would be impossible to subsume a clause that does not contain the relation.

The algorithm operates by iteratively building hypotheses. For some arbitrary clause $C \in Ex^+$, the algorithm will try every signed relation name in the clause as a potential guard (line 2). For each such candidate, the lgig induced by Ex^+ is generated and added to a candidate hypothesis H. From this point on the algorithm will try to add further literals from C to H via their lgig map as long as they do not break subsumption of Ex^+. This either continues until H is identified as a solution (if also $H \not\vdash Ex^-$), or until all possible additions are exhausted. If no solution is found for any choice of G, then the algorithm rejects.

It is straightforward to observe the soundness of Algorithm 1. By definition, any returned clause H is a guarded clause with $H \vdash Ex^+$ and $H \not\vdash Ex^-$. What is left to show is that the algorithm is complete, i.e., that it will always return a hypothesis if the instance has a solution, and that it is tractable. Completeness follows from the observations on lgig maps made above and Theorem 9, showing that if a solution exists, then a straight solution guarded by a lgig exists. A full argument is given in the technical appendix.

Theorem 10. *Algorithm 1 is a correct algorithm for the ILP consistency problem for $\mathcal{G}^*$.*

The worst-case time complexity of Algorithm 1 depends on the complexity of computing the lgig, the lgig maps, and checking subsumption. Computing the lgig for literal P is effectively in $O(\mathrm{ar}(P)^2 \cdot |Ex^+|)$, since it only requires the identification of positions that contain equal terms in all clauses. Similarly, from Lemma 7 we see that the lgig map for a literal L can be constructed directly from the respective relative shields. Again this can be done in time $O(\mathrm{ar}(L)^2 \cdot |Ex^+|)$, followed by a lookup of the relevant positions in the guard literal. As stated previously in Lemma 4, checking subsumption is indeed tractable for guarded formulas. Clearly, the hypothesis is guarded in all subsumption checks in Algorithm 1 and therefore all of the checks are in polynomial time.

Theorem 11. *The ILP consistency problem for $\mathcal{G}^*$ is in* PTIME.

Enumerating representative solutions While the decision problem is important for theoretical considerations, in practice the enumeration of solutions is of greater interest. In our setting $\mathcal{G}^*$, it is still possible to have exponentially many solutions. We propose considering the (subset) maximal lgig guarded solutions as the *canonical solutions* of the problem. Note that there are only polynomially many such solutions.

This definition is motivated by Theorem 9 and the preceding lemmas, which show that every solution is either guarded by a lgig or can be specialised to a solution guarded by a lgig. In this sense, our canonical solutions are representative solutions of the problem. It is then straightforward to adapt Algorithm 1 to instead enumerate all of the canonical solutions in polynomial time.

Guarded rules in Datalog$^\pm$ Guarded rules are particularly popular in Datalog$^\pm$, where they guarantee decidability for query answering. Recall, a rule in Datalog$^\pm$ can contain existential quantification in the head. For example, the following Datalog $^\pm$ rule can express that every person x has an ancestor y.

$$\exists y Ancestor(y, x) \leftarrow Person(x)$$

Hence, the notion of guardedness of a rule is adapted such that the guard must cover only free variables.

This is not an issue for our algorithm. The analogue to learning a guarded Datalog$^\pm$ rule in our setting, is to learn Horn clauses where only the negative literals must be guarded. Any unguarded variables in the positive literal then correspond to existentially quantified variables. By adapting the notion of lgig map accordingly for positive atoms (positions with no shield are mapped to fresh variables) and distinguishing positive and negative literals, Algorithm 1 can easily be adapted to a polynomial time algorithm for learning guarded Datalog$^\pm$ rules.

Generalisation to k-Guardedness

We have shown that the restriction to guarded hypotheses greatly reduces the complexity of the ILP consistency problem. While guarded clauses have been repeatedly shown to exhibit a rare balance between expressiveness and low complexity for reasoning, some important concepts are not expressible in guarded form. One particularly important such case is transitive closure, e.g., we can construct the transitive closure T over a relation R for example with the rule

$$T(x, y), R(y, z) \rightarrow T(x, z)$$

but not in terms of a guarded rule.

However, the clause is clearly 2-guarded, as are many other common logical properties and – from anecdotal observations – as are many rules derived from expert knowledge in industry. In this section we will briefly discuss how our results from the previous sections can be generalised to also hold for k-guarded clauses, where k is constant.

Overall, no major changes are necessary to accommodate k-guardedness. Our observations on guarded hypotheses for straight clauses rely on the fact that the variables in the guard have a uniquely-determined assignment in any clause that is subsumed by the guarded hypothesis. This observation can be extended to the case of a set of k literals that k-guard the hypothesis by viewing them as one *merged* literal G whose argument list is simply the concatenation of the individual argument lists. That is, in the transitive closure rule above we could consider the merged guard $G(x, y, y, z)$ instead of the literals $\{\neg T(x, y), negR(y, z)\}$ that guard the respective clause. It is straightforward to verify that such a merged guard behaves exactly the same as any normal guard literal.

There are $\binom{|C|}{k} \leq |C|^k$ possible ways to replace k literals in a clause C with such a merged guard. Hence, an adapted Algorithm 1 for k-guarded hypothesis would have to try only $|C|^k$ possible guard candidates. The computation of the lgig and lgig maps is also still polynomial since the arity of the merged guard is simply the sum of the individual literal arities. By Lemma 4 the subsumption checks remain tractable under k-guardedness. We therefore can also state the following more general tractability result.

Theorem 12. *Fix an integer k. Let $\mathcal{G}^*_k$ be the ILP setting like $\mathcal{G}^*$ where the hypothesis language is relaxed to allow k-guarded clauses. Then the ILP consistency problem for $\mathcal{G}^*_k$ is in* PTIME.

Conclusion & Outlook

We have studied the complexity of the ILP consistency problem when learning guarded clauses. In general, the problem is NP-complete, in contrast to the case of learning Horn clauses with Horn examples, which is Σ_2^P-complete. We show how further restriction of positive examples allows for efficient learning of guarded clauses. The tractability result also extends to the even more general class k-guarded rules. An important question that was left open is the integration of background knowledge. Once we allow background knowledge with variables we are confronted with conceptual issues on whether and how the guardedness constraint should extend to background knowledge or not.

Acknowledgements

Andrei Draghici is supported by a grant from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT). Georg Gottlob is a Royal Society Research Professor and acknowledges support by the Royal

Society in this role through the "RAISON DATA" project (Reference No. RP\R1\201074). Matthias Lanzinger acknowledges support by the Royal Society "RAISON DATA" project (Reference No. RP\R1\201074).

References

Alphonse, É.; and Osmani, A. 2009. Empirical Study of Relational Learning Algorithms in the Phase Transition Framework. In *Proc. ECML PKDD*, volume 5781 of *Lecture Notes in Computer Science*, 51–66. Springer.

Bellomarini, L.; Sallinger, E.; and Gottlob, G. 2018. The Vadalog System: Datalog-based Reasoning for Knowledge Graphs. *Proc. VLDB Endow.*, 11(9): 975–987.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2009. A general Datalog-based framework for tractable query answering over ontologies. In *Proc. PODS*, 77–86. ACM.

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Semant.*, 14: 57–83.

Cohen, W. W.; and Jr., C. D. P. 1995. Polynomial Learnability and Inductive Logic Programming: Methods and Results. *New Gener. Comput.*, 13(3&4): 369–409.

Cropper, A.; Dumancic, S.; and Muggleton, S. H. 2020. Turning 30: New Ideas in Inductive Logic Programming. In *Proc. IJCAI*, 4833–4839. ijcai.org.

Fitting, M. 1996. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer. ISBN 978-1-4612-7515-2.

Fürnkranz, J.; Gamberger, D.; and Lavrac, N. 2012. *Foundations of Rule Learning*. Cognitive Technologies. Springer. ISBN 978-3-540-75196-0.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman. ISBN 0-7167-1044-7.

Gottlob, G.; Grädel, E.; and Veith, H. 2002. Datalog LITE: a deductive query language with linear time model checking. *ACM Trans. Comput. Log.*, 3(1): 42–79.

Gottlob, G.; Leone, N.; and Scarcello, F. 1997. On the Complexity of Some Inductive Logic Programming Problems. In *Proc. ILP*, volume 1297 of *Lecture Notes in Computer Science*, 17–32. Springer.

Gottlob, G.; Leone, N.; and Scarcello, F. 2002. Hypertree Decompositions and Tractable Queries. *J. Comput. Syst. Sci.*, 64(3): 579–627.

Grohe, M. 2007. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1): 1:1–1:24.

Kietz, J.; and Dzeroski, S. 1994. Inductive Logic Programming and Learnability. *SIGART Bull.*, 5(1): 22–32.

Kietz, J.; and Lübbe, M. 1994. An Efficient Subsumption Algorithm for Inductive Logic Programming. In *Proc. Machine Learning*, 130–138. Morgan Kaufmann.

Muggleton, S. 1991. Inductive Logic Programming. *New Gener. Comput.*, 8(4): 295–318.

Muggleton, S.; and Feng, C. 1990. Efficient Induction of Logic Programs. In *Proc. ALT*, 368–381. Springer/Ohmsha.

Papadimitriou, C. H. 2007. *Computational complexity*. Academic Internet Publ. ISBN 978-1-4288-1409-7.

Raghothaman, M.; Mendelson, J.; Zhao, D.; Naik, M.; and Scholz, B. 2020. Provenance-guided synthesis of Datalog programs. *Proc. ACM Program. Lang.*, 4(POPL): 62:1–62:27.

Si, X.; Raghothaman, M.; Heo, K.; and Naik, M. 2019. Synthesizing Datalog Programs using Numerical Relaxation. In *Proc. IJCAI*, 6117–6124. ijcai.org.

Ullman, J. D. 1989. *Principles of Database and Knowledge-Base Systems, Volume II*. Computer Science Press. ISBN 0-7167-8162-X.