Meta-interpretive Learning: application to Grammatical Inference

Stephen H. Muggleton, Dianhuan Lin, Niels Pahlavi and Alireza Tamaddoni-Nezhad

Department of Computing, Imperial College London

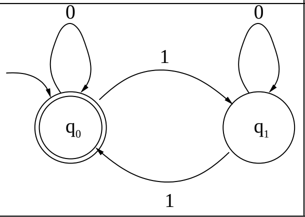
Abstract. Despite early interest Predicate Invention has lately been under-explored within ILP. We develop a framework in which predicate invention and recursive generalisations are implemented using abduction with respect to a meta-interpreter. The approach is based on a previously unexplored case of Inverse Entailment for Grammatical Inference of Regular languages. Every abduced grammar H is represented by a conjunction of existentially quantified atomic formulae. Thus $\neg H$ is a universally quantified clause representing a denial. The hypothesis space of solutions for $\neg H$ can be ordered by θ-subsumption. We show that the representation can be mapped to a fragment of Higher-Order Datalog in which atomic formulae in H are projections of first-order definite clause grammar rules and the existentially quantified variables are projections of first-order predicate symbols. This allows predicate invention to be effected by the introduction of first-order variables. This application of abduction to conduct predicate invention is related to that of previous work by Inoue and Furukawa. We show that the approach is sufficiently flexible to support learning of Context-Free grammars from positive and negative example, a problem shown to be theoretically possible by E.M. Gold in the 1960s, though to the authors' knowledge, not previously demonstrated within Grammatical Inference. We describe the implementation of Metagol$_R$ and Metagol$_{CF}$ for learning Regular and Context-Free grammars respectively. Experiments indicate that on randomly chosen grammars both systems run significantly faster (100 to 1000 times) than the state-of-the-art ILP system MC-TopLog and have significantly higher predictive accuracies than MC-TopLog. Lastly we demonstrate that by combining Metagol$_R$ and Metagol$_{CF}$ we can formulate a system, Metagol$_{RCF}$, which can change representation by firstly assuming the target to be Regular, and then failing this, switch to assuming it to be Context-Free. Once again, Metagol$_{RCF}$ runs up to 100 times faster than Metagol$_{CF}$ on grammars chosen randomly from Regular and non-Regular Context-Free grammars. In conclusion we discuss ways in which the approach could be extended to predicate invention in fragments of first-order logic other than grammars.

1 Introduction

Consider the problem of using an ILP system to learn a Regular grammar which accepts all and only those binary sequences containing an even number of $1s$ (see Figure 1). It has been shown [12] that this so-called *Parity* grammar is unlearnable for decision tree learning algorithms applied to k-length sequences. By contrast, since the 1950s automaton-based learning algorithms have existed [11] which inductively infer Regular languages, such as *Parity*, from positive and negative examples. If we try to learn *Parity* using an ILP system the obvious representation of the target would be a Definite Clause Grammar (DCG) (see Figure 1a). However, if the ILP system were provided with examples for the predicate q_0 then the predicate q_1 would need to be invented. It is widely accepted that Predicate Invention is a hard and under-explored topic within ILP [18], and indeed state-of-the-art ILP systems, including MC-TopLog [17] and Progol [13, 15], are unable to learn grammars such as *Parity* in the form of a DCG since these systems do not support Predicate Invention. However, note that in Figure 1a each clause of the DCG has one of the following two forms.

$$Q([], []) \leftarrow$$
$$Q([C|x], y) \leftarrow P(x, y)$$

where Q, C, P are the only symbols which vary between the clauses. Figure 1b shows how these two forms of clauses above can be captured within the two clauses of a recursive meta-interpreter *parse/3* a)

Finite acceptor	**Production rules**	**Definite Clause Grammar (DCG)**	**Positive examples**	**Negative examples**
			λ	1
	$q_0 \rightarrow$	$q_0([],[]) \quad \leftarrow$	0	01
	$q_0 \rightarrow 0\ q_0$	$q_0([0\|A],B) \leftarrow q_0(A,B)$	00	10
	$q_0 \rightarrow 1\ q_1$	$q_0([1\|A],B) \leftarrow q_1(A,B)$	11	001
	$q_1 \rightarrow 0\ q_1$	$q_1([0\|A],B) \leftarrow q_1(A,B)$	000	010
	$q_1 \rightarrow 1\ q_0$	$q_1([1\|A],B) \leftarrow q_0(A,B)$	011	100
			101	111

b)

Meta-interpreter	**Ground facts**
$parse(S) \leftarrow parse(q0, S, [])$.	$acceptor(q0) \leftarrow$
	$delta1(q0, 0, q0) \leftarrow$
$parse(Q, [], []) \leftarrow acceptor(Q)$.	$delta1(q0, 1, q1) \leftarrow$
$parse(Q, [C\|X], Y) \leftarrow delta1(Q, C, P),\ parse(P, X, Y)$.	$delta1(q1, 0, q1) \leftarrow$
	$delta1(q1, 1, q0) \leftarrow$

Fig. 1. a) Parity acceptor with associated production rules, DCG, positive and negative examples; b) Meta-interpreter with ground facts representing Parity grammar

which uses the auxilliary predicates *acceptor/1* and *delta1/3*[1][2] to instantiate the predicate symbols and constants from the original DCG. The predicates *acceptor/1* and *delta1/3* can each be interpreted as Higher-Order Datalog [20] predicates since they take arguments which are predicate symbols q_0, q_1 from the DCG. By making *acceptor/1* and *delta1/3* abducible, *Parity*, and indeed any other Regular grammar, could in principle be learned from ground instances of *parse/1* using abduction. The paper explores this form of learning with respect to a meta-interpreter.

We show that such abductively inferred grammars are a special case of Inverse Entailment. We also show that the hypothesis space forms a lattice ordered by subsumption. The extensions of this use of abduction with respect to a meta-interpreter lead to a new class of inductive algorithm for learning Regular and Context-Free languages. The new approach blurs the normal distinctions between abductive and inductive techniques (see [5]). Usually abduction is thought of as providing an explanation of a single ground fact in the form of a set of ground facts while induction provides an explanation of a set of ground facts in the form of a set of universally quantified rules. However, the meta-interpreter in Figure 1b can be viewed as projecting the universally quantified rules in Figure 1a onto the ground facts associated with *acceptor/1* and *delta1/3* in Figure 1b. In this way abducing these ground facts with respect to a meta-interpreter is equivalent to induction, since it is trivial to map the ground *acceptor/1* and *delta1/3* facts back to the original universally quantified DCG rules.

The paper is structured as follows. Section 2 introduces the theoretical framework for Meta-interpretive learning and its application to grammatical inference. We then describe a system implementation for three related systems, Metagol$_R$, Metagol$_{CF}$ and Metagol$_{RCF}$ in Section 3. In Section 4 the performance of these systems is compared experimentally against MC-Toplog on Regular and

[1] Note that in the theory of automata [8] *delta1/3* corresponds to the transition function of the finite acceptor shown in Figure 1a.

[2] Considering *delta1/3* as an arity 3 ground relation, if c, k are bounds on the number of terminals and non-terminals respectively then the number of possible definitions for *delta1/3* is 2^{ck^2}.

Context-Free grammar learning problems. In Section 5 we describe related work. Lastly we conclude and describe directions for further work in Section 6.

2 Meta-interpretive Learning framework

2.1 Logical notation

A variable is represented by an upper case letter followed by a string of lower case letters and digits. A function symbol is a lower case letter followed by a string of lower case letters and digits. A predicate symbol is a lower case letter followed by a string of lower case letters and digits. The arity of a function or predicate symbol is the number of arguments it takes. A constant is a function or predicate symbol which has arity zero. Variables and constants are terms, and a function symbol immediately followed by a bracketed n-tuple of terms is a term. Thus $f(g(X), h)$ is a term when f, g and h are function symbols and X is a variable. A predicate symbol immediately followed by a bracketted n-tuple of terms is called an atomic formula. The negation symbol is: $\neg$. Both A and $\neg A$ are literals whenever A is an atomic formula. In this case A is called a positive literal and $\neg A$ is called a negative literal. A finite set (possibly empty) of literals is called a clause. A clause represents the disjunction of its literals. Thus the clause $\{A_1, A_2, ..\neg A_i, \neg A_{i+1}, ...\}$ can be equivalently represented as $(A_1 \vee A_2 \vee ..\neg A_i \vee \neg A_{i+1} \vee ...)$ or $A_1, A_2, .. \leftarrow A_i, A_{i+1}, ...$ A Horn clause is a clause which contains at most one positive literal. A Horn clause is unit if and only if it contains exactly one literal. A denial or goal is a Horn clause which contains no positive literals. A definite clause is a Horn clause which contains exactly one positive literal. The positive literal in a definite clause is called the head of the clause while the negative literals are collectively called the body of the clause. A unit clause is positive if it contains a head and no body. A unit clause is negative if it contains one literal in the body. A set of clauses is called a clausal theory. A clausal theory represents the conjunction of its clauses. Thus the clausal theory $\{C_1, C_2, ...\}$ can be equivalently represented as $(C_1 \wedge C_2 \wedge ...)$. A clausal theory in which all predicates have arity at most one is called monadic. A clausal theory in which all predicates have arity at most two is called diadic. A clausal theory in which each clause is Horn is called a logic program. A logic program is said to be definite in the case it contains only definite clauses. A logic program is said to be a Datalog program if it contains no function symbols other than constants. A Datalog program is said to be Higher-Order in the case that it contains at least one constant predicate symbol which is the argument of a term. Literals, clauses and clausal theories are all well-formed-formulae (wffs) in which the variables are assumed to be universally quantified. Let E be a wff or term. E is said to be ground if and only if it contains no variables. Let C and D be clauses. We say that $C \succeq_\theta D$ or C θ-subsumes D if and only if there exists a substitution θ such that $C\theta \subseteq D$.

2.2 Formal language notation

Let Σ be a finite alphabet. Σ^* is the infinite set of strings made up of zero or more letters from Σ. λ is the empty string. uv is the concatenation of strings u and v. $|u|$ is the length of string u. A language L is any subset of Σ^*. Let ν be a set of non-terminal symbols disjoint from Σ. A production rule $r = LHS \rightarrow RHS$ is well-formed in the case that $LHS \in (\nu \cup \Sigma)^*$, $RHS \in (\nu \cup \Sigma \cup \lambda)^*$ and when applied replaces LHS by RHS in a given string. A grammar G is a pair $\langle s, R\rangle$ consisting of a start symbol $s \in \nu$ and a finite set of production rules R. A grammar is Regular Chomsky-normal in the case that it contains only production rules of the form $S \rightarrow \lambda$ or $S \rightarrow aB$ where $S, B \in \nu$ and $a \in \Sigma$. A grammar is Linear Context-Free in the case that it contains only Regular Chomsky-normal production rules or rules of the form $S \rightarrow Ab$ where $S, A \in \nu$ and $b \in \Sigma$. A grammar is Context-Free in the case that it contains only Linear Context-Free Chomsky-normal production rules or rules of the form $S \rightarrow AB$ where $S, A, B \in \nu$.[3] A Context-Free grammar is said to be deterministic in the case that it does not contain two Regular Chomsky-normal production rules $S \rightarrow aB$ and $S \rightarrow aC$ where $B \neq C$. A sentence $\sigma \in \Sigma^*$ is in $L(G)$ iff given a start symbol $S \in \nu$ there exists a sequence of production rule applications $S \rightarrow_{R_1} \ldots \rightarrow_{R_n} \sigma$ where $R_i \in G$. A language L is Regular, Linear Context-free or Context-Free in the case there exists a grammar G for which $L = L(G)$ where G is Regular, Linear Context-Free or Context-Free respectively. According to the Context-Free Pumping Lemma, if a language L is Context-Free, then there exists some integer $p \geq 1$ such that any string s in L with $|s| \geq p$ (where p is a pumping length) can be written as $s = uvxyz$ with substrings u, v, x, y and z, such that $|vxy| \leq p$, $|vy| \geq 1$ and uv^nxy^nz is in L for every integer $n \geq 0$.

2.3 Framework

The Meta-Interpretive Learning (MIL) setting is a variant of the normal setting for ILP.

Definition 1 (Meta-Interpretive Learning setting). *A Meta-Interpretive Learning (MIL) problem consists of Input = $\langle B, E\rangle$ and Output = $H \in \mathcal{H}_{B,E}$ where the background knowledge $B = \langle B_M, B_A\rangle$. B_M is a logic program representing a meta-interperter and B_A and H are ground definite Higher-Order Datalog programs consisting of positive unit clauses. The predicate symbol constants in B_A and H are represented by Skolem constants. The examples are $E = \langle E^+, E^-\rangle$ where E^+ is a ground logic program consisting of positive unit clauses and E^- is a ground logic program consisting of negative unit clauses. The Input and Output are such that $B, H \models E^+$ and for all e^- in E^-, $B, H \not\models e^-$.*

Inverse Entailment can be applied to allow H to be derived from B and E^+ as follows.

$$\begin{aligned} B, H &\models E^+ \\ B, \neg E^+ &\models \neg H \end{aligned} \qquad (1)$$

Since both H and E^+ can each be treated as conjunctions of ground atoms containing Skolem constants in place of existential variables, it follows that $\neg H$ and $\neg E^+$ are universally quantified denials where the variables come from replacing Skolem constants by unique variables. We now define the concept of a Meta-interpretive learner.

Definition 2 (Meta-interpretive learner). *Let $\mathcal{H}_{B,E}$ represent the complete set of abductive solutions for the MIL setting of Definition 1. Algorithm A is said to be a Meta-interpretive learner iff for all B, E such that $H = A(B, E)$ it is the case that $H \in \mathcal{H}_{B,E}$.*

Example 1 (Parity example). Let $B = \langle B_M, B_A\rangle$, $E = \langle E^+, E^-\rangle$ and $H \in \mathcal{H}_{B,E}$ represents the parity grammar. Figure 2 shows H as a possible output of a Meta-interpretive learner.

Note that in this example abduction with respect to B_M produces Predicate Invention by introducing Skolem constants representing new predicate symbols. By contrast an ILP system such as Progol uses Inverse Entailment [13] to construct a single clause from a single example, while a Meta-interpretive learner uses Inverse Entailment to construct the set of all clauses H as the abductive solution to a single goal $\neg E^+$ using E^- as integrity constraints. In the example the hypothesised grammar H correponds to the first-order DCG from Figure 1a, which contains both invented predicates and mutual recursion. Neither predicate invention nor mutual recursion can be acheived with DCGs in this way using ILP systems such as Progol or MC-TopLog.

[3] This is an adaptation of Chomsky-normal form Context-free, which usually only permits productions of the form $S \rightarrow \lambda$, $S \rightarrow a$ and $S \rightarrow AB$.

E^+	$\neg E^+$	E^-	H	$\neg H$
$parse([]) \leftarrow$	$\leftarrow parse([]),$	$\leftarrow parse([1])$	$acceptor(\$0) \leftarrow$	$\leftarrow acceptor(Q0),$
$parse([1,1]) \leftarrow$	$parse([1,1]),$	$\leftarrow parse([0,1])$	$delta1(\$0,0,\$0) \leftarrow$	$delta1(Q0,0,Q0),$
$parse([0,1,1]) \leftarrow$	$parse([0,1,1]),$	$\leftarrow parse([1,0])$	$delta1(\$0,1,\$1) \leftarrow$	$delta1(Q0,1,Q1),$
$parse([1,0,1]) \leftarrow$	$parse([1,0,1]),$	$\leftarrow parse([0,0,1])$	$delta1(\$1,0,\$1) \leftarrow$	$delta1(Q1,0,Q1),$
$parse([1,1,0]) \leftarrow$	$parse([1,1,0]).$	$\leftarrow parse([1,1,1])$	$delta1(\$1,1,\$0) \leftarrow$	$delta1(Q1,1,Q0).$

Fig. 2. Parity example where B_M is the Meta-interpreter shown in Figure 1b, $B_A = E^- = \emptyset$ and E^+, $\neg E^+$, H, $\neg H$, are as shown above. '\$0' and '\$1' in H is a Skolem constant replacing existentially quantified variables.

2.4 Lattice properties of hypothesis space

In this section we investigate orderings over MIL hypotheses.

Definition 3 ($\succeq_{B,E}$ relation in MIL). *Within the MIL setting we say that $H \succeq_{B,E} H'$ in the case that $H, H' \in \mathcal{H}_{B,E}$ and $\neg H' \succeq_\theta \neg H$.*

We now show that $\succeq_{B,E}$ forms a quasi-ordering and a lattice.

Proposition 1 (Quasi-ordering). *Within the MIL setting $\langle \mathcal{H}_{B,E}, \succeq_{B,E} \rangle$ forms a quasi-ordering.*

Proof. *Follows from the fact that $\langle \{\neg H : H \in \mathcal{H}_{B,E}\}, \succeq_\theta \rangle$ forms a quasi-ordering since each $\neg H$ is a clause [21].*

Proposition 2 (Lattice). *Within the MIL setting $\langle \mathcal{H}_{B,E}, \succeq_{B,E} \rangle$ forms a lattice.*

Proof. *Follows from the fact that $\langle \{\neg H : H \in \mathcal{H}_{B,E}\}, \succeq_\theta \rangle$ forms a lattice since each $\neg H$ is a clause [21].*

We now show that this ordering has a unique top element.

Proposition 3 (Unique $\top$ element). *Within the MIL setting there exists $\top \in \mathcal{H}_{B,E}$ such that for all $H \in \mathcal{H}_{B,E}$ we have $\top \succeq_{B,E} H$ and $\top$ is unique up to renaming of Skolem constants.*

Proof. *Let $\neg H' = \bigvee_{H \in \mathcal{H}_{B,E}} \neg H$ and $\neg\top = \neg H'\theta_v$ where v is a variable and $\theta_v = \{u/v : u \text{ variable in } \neg H'\}$. By construction for each $H \in \mathcal{H}_{B,E}$ it follows that $\neg\top \succeq_\theta \neg H$ with subsitution θ_v. Therefore for all $H \in \mathcal{H}_{B,E}$ we have $\top \succeq_{B,E} H$ and $\top$ is unique up to renaming of Skolem constants.*

This proposition can be illustrated with a grammar example.

Example 2 (Subsumption example). In terms of the Meta-interpreter of Figure 1a the universal grammar $\{0,1\}^*$ can be expressed using $\top = \{(acceptor(\$0) \leftarrow), (delta1(\$0,0,\$0) \leftarrow), (delta1(\$0,1,\$0) \leftarrow)\}$. Letting H represent the Parity grammar from Example 1 it is clear that $\neg H \succeq_\theta \neg\top$ and so $\top \succeq_{B,E} H$. So unlike the subsumption relation between universally quantified clauses, binding all the (existentially quantified) variables in H to each other produces a maximally general grammar $\top$.

We now show the conditions under which there is a unique bottom element of the lattice.

Proposition 4 (Unique $\bot$ element). *In the case that $\mathcal{H}_{B,E}$ is finite up to renaming of Skolem constants there exists $\bot \in \mathcal{H}_{B,E}$ such that for all $H \in \mathcal{H}_{B,E}$ we have $H \succeq_{B,E} \bot$ and $\bot$ is unique up to renaming of Skolem constants.*

Proof. *Since $\mathcal{H}_{B,E}$ is finite $\neg\bot = lgg(\{\neg H : H \in \mathcal{H}_{B,E}\})$ where lgg is Plotkin's algorithm for computing the least general generalisation of a set of clauses under subsumption [22].*

For most purposes the construction of the unique bottom clause is intractable since the cardinality of the lgg clause increases exponentially in the cardinality of $\mathcal{H}_{B,E}$. We now show a method for reducing hypotheses.

Language type	**Meta-interpreter**	**Example Grammar**	**Example Language**
a) Regular	$parse(S) \leftarrow parse(Q, S, [])$. $parse(Q, X, X) \leftarrow acceptor(Q)$. $parse(Q, [C\|X], Y) \leftarrow delta1(Q, C, P), parse(P, X, Y)$.	$S \rightarrow 0\ S$ $S \rightarrow 1\ T$ $T \rightarrow \lambda$ $T \rightarrow 1\ T$	0^+1^+
b) Context-Free	$parse(S) \leftarrow parse(Q, S, [])$. $parse(Q, X, X) \leftarrow acceptor(Q)$. $parse(Q, [C\|X], Y) \leftarrow delta1(Q, C, P), parse(P, X, Y)$. $parse(Q, X, Y) \leftarrow delta2(Q, P, C), parse(P, X, [C\|Y])$. $parse(Q, X, Y) \leftarrow delta3(Q, P, R), parse(P, X, Z), parse(R, Z, Y)$.	$S \rightarrow \lambda$ $S \rightarrow T\ S$ $T \rightarrow 0\ U$ $U \rightarrow T\ 1$	$(0^n1^n)^*$

Fig. 3. Meta-interpreters, example Chomsky-normal form grammars and example languiages for a) Regular and b) Context-Free languages

2.5 Reduction of hypotheses

Proposition 5 (Logical reduction of hypotheses.). *Suppose H' is an hypothesis in the MIL setting and $\neg H$ is the result of applying Plotkin's clause reduction algorithm [22] to $\neg H'$. Then H is a reduced hypothesis equivalent to H'.*

Proof. *Follows from the fact that $\neg H'$ is θ-subsumption equivalent to $\neg H$ by construction.*

Example 3 (Reduction example). Let $H' = H \cup \{r\}$ where H is the Parity grammar from Figure 2 and $r = (delta1(\$0, 0, \$2) \leftarrow)$ represents an additional redundant grammar rule. Now Plotkin's reduction algorithm would reduce $\neg H'$ to the equivalent clause $\neg H$ and consequently grammar H is a reduced equivalent form of H.

In the following section we show the existence of a compact bottom hypothesis in the case of MIL for Regular languages.

2.6 Framework applied to grammar learning

Figure 3 shows how the Meta-interpreter for Regular Grammars, can be extended to Context-Free Grammars. The Chomsky language types form an inclusion hierarchy in which Regular $\subseteq$ Context-Free. Algorithms for learning the Regular languages have been widely studied since the 1970s within the topic of Grammatical Inference [3]. Many of these start with a prefix tree acceptor, and then progressively merge the states.

Proposition 6 (Unique $\bot$ for Regular languages). *Prefix trees act as a compact bottom theory in the MIL setting for Regular languages.*

Sketch Proof. *Follows from the fact that all deterministic Regular gramamrs which include the positive examples can be formed by merging the arcs of a prefix tree acceptor [12]. Merging the arcs of the prefix tree is achieved by unifying the delta1 atoms in $\neg H$ within the MIL setting.*

Example 4 (Prefix tree). Assume the MIL setting with B_M being the meta-interpreter for Regular languages. Let $E^+ = \{parse([1,1]), parse([1,1,0])\}$ then $\bot = \{delta1(\$0, 1, \$1), delta1(\$1, 1, \$2), acceptor(\$2), delta1(\$2, 0, \$3), acceptor(\$3)\}$ represents the prefix tree automaton.

Proposition 7 ($\bot$ for Context-Free languages). *Any bottom theory $\bot$ for a Context-Free language contains a set of $delta1$ atoms representing a Regular prefix tree.*

Sketch Proof. *Follows from the fact that the Regular subset of MIL hypotheses are all subsumed by $\neg\bot_R$ where $\bot_R$ represents the Regular prefix tree.*

3 Implementations

The systems Metagol$_R$, Metagol$_{CF}$ and Metagol$_{RCF}$ are three simple Prolog implementations of Meta-interpretive learning.

Metagol$_R$ The Metagol$_R$ system is based on the following abductive variant of the Regular Meta-interpreter from Figure 3 (the standard definition of member/2 is omitted for brevity).

```
parse(S,G1,G2) :- parse(s(0),S,[],G1,G2).

parse(Q,X,X,G1,G2) :- abduce(acceptor(Q),G1,G2).
parse(Q,[C|X],Y,G1,G2) :- skolem(P), abduce(delta1(Q,C,P),G1,G3), parse(P,X,Y,G3,G2).

abduce(X,G,G) :- member(X,G).
abduce(X,G,[X|G]) :- not(member(X,G)).

skolem(s(0)). skolem(s(1)). ...
```

The abduced atoms are simply accumlulated in the extra variables $G1, G2, G3$. The term $s(0)$ represents the start symbol and a finite set of Skolem constants is provided by the monadic predicate *skolem*. Hypotheses are now the answer substitutions of a goal such as the following.

```
:- parse([],[],G1), parse([0],G1,G2), parse([0,0],G2,G3), parse([1,1],G3,G4), % Positives
   parse([0,0,0],G4,G5), parse([0,1,1],G5,G6), parse([1,0,1],G6,G),
   not(parse([1],G,G)), not(parse([0,1],G,G)).                                 % Negatives
```

Note that each of the positive examples are provided sequentially within the goal and the resulting grammar is then tested for non-coverage of each of the negative examples. In the case shown above the first hypothesis found by Prolog is as follows.

```
G = [delta1(s(1),0,s(1)),delta1(s(1),1,s(0)),delta1(s(0),1,s(1)),delta1(s(0),0,s(0)),acceptor(s(0))]
```

This hypothesis correctly represents the Parity acceptor of Figure 1. All other consistent hypotheses can be generated by making Prolog backtrack through the SLD proof space.

Metagol$_{CF}$ The Metagol$_{CF}$ system is based on an abductive variant of the Context-Free Meta-interpreter from Figure 3, though we omit the full Prolog description due to space restrictions. Once more, abduction is carried out with respect to a single goal as in Metagol$_R$. However, in order to increase efficiency the goal first abduces a skolemised bottom theory for a seed example. This bottom theory is a special case instance of a prefix tree acceptor (see Proposition 7). The remainder of the goal uses the meta-interpreter to abduce a set of additional productions which satisfy the remaining positive and negative examples.

Metagol$_{RCF}$ The Metagol$_{RCF}$ system simply combines Metagol$_R$ and Metagol$_{CF}$ sequentially. Thus the hypothesis returned will be Regular in the case Metagol$_R$ finds a consistent grammar and otherwise will be context-free if Metagol$_{CF}$ finds a consistent grammar.

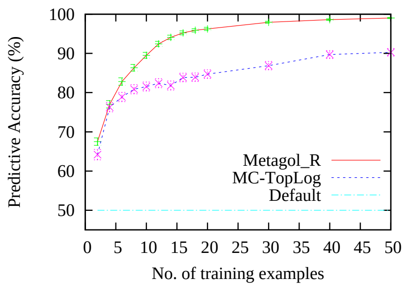
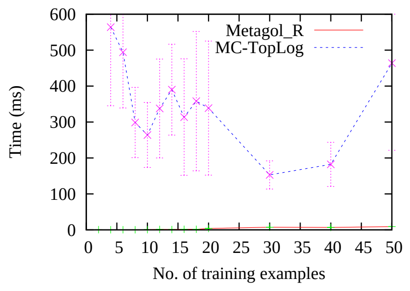
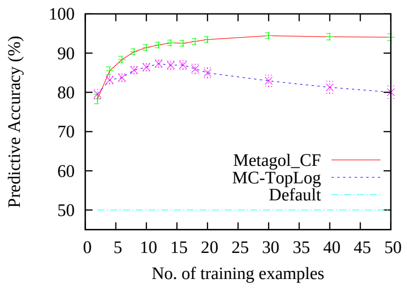
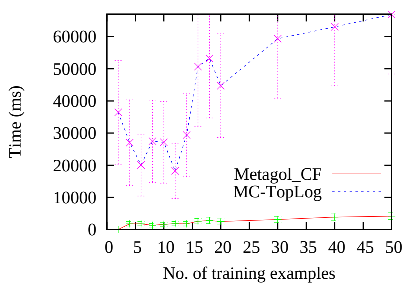
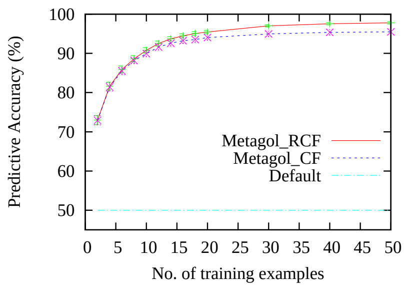
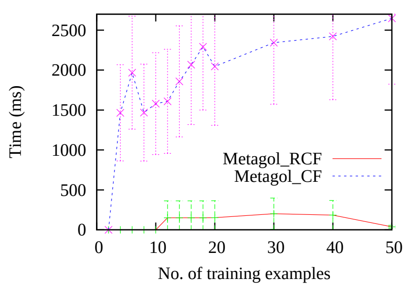
	Predictive accuracy	**Running time**
a) Regular Null hypothesis 1		
b) Context-Free Null hypothesis 2		
c) Representation change Null hypothesis 3		

Fig. 4. Average predictive accuracies and running times for Null hypotheses 1,2 and 3

4 Experiments

In this section we describe experiments on learning Regular and Context-Free grammars[4]. It was shown in Section 1 that ILP systems cannot learn grammars in a DCG representation with predicate invention. However, an ILP system given a meta-interpreter as background knowledge can be used to learn within the MIL framework. In the experiments described below we compare the performance of a state-of-the-art ILP system MC-TopLog, loaded with suitable meta-interpretive background against the simple Prolog MIL implementations described in Section 3.

4.1 Learning Regular Languages

Null Hypothesis 1.1 Metagol$_R$ cannot learn randomly chosen Regular languages.

[4] The Metagol code and datasets for these experiments will be made available on a website associated with the final paper

Null Hypothesis 1.2 Metagol$_R$ cannot outperform a state-of-the-art ILP system on learning randomly chosen Regular languages.

Materials and Methods Randomly chosen Regular grammars were generated by sampling from a Stochastic Logic Program (SLP) [14] which defined the space of grammars. Each randomly generated grammar was reduced using the Plotkin reduction algorithm (see Section 2.5) in order to remove redundancy and equivalent non-terminals.

A randomly generated grammar was discarded if it was non-deterministic or corresponded to a finite language. The maximum number of non-terminals allowed in this experiment was 5. The examples were randomly chosen from Σ^* for $\Sigma = \{a, b\}$. The maximum length of sequences considered in this experiment was 15. The sampling of examples was also done using an SLP. Sampling was conducted with replacement, therefore there were duplicates among the sampled examples and shorter sequences had higher probabilities. The label of each example was decided by querying against the sampled grammar and background knowledge. We used MC-TopLog with Regular Meta-interpreter background knowledge against Metagol$_R$.

We tested Metagol$_R$ and MC-TopLog's performance on learning Regular grammars using 200 different randomly chosen Regular grammars. Their performance was evaluated by the predictive accuracies and running time. The results were averaged over the 200 sampled grammars. For each sample, the learning curve was obtained by varying the size of training set from 2 to 50, while a fixed test set of size 1000 was used for evaluating predictive accuracy.

Results and Discussion As shown by Figure 4a, the predictive accuracy of Metagol$_R$ increases with increasing numbers of training examples, and it approaches 100% when the size of training set is larger than 25. Therefore Null hypothesis 1.1 is refuted. The Metagol$_R$ accuracy is signficantly higher than that of MC-TopLog. However, MC-TopLog's running time is significantly longer (around 100 times) than that of Metagol$_R$ as shown in Figure 4a. This is because MC-TopLog does not have an ordered search space so that it has to enumerate all candidate hypotheses within the version space. In contrast, Metagol$_R$ does a bounded search. Null hypothesis 1.2 is also refuted with respect to both predictive accuracy and running time.

4.2 Learning Context-Free Languages

Null Hypothesis 2.1 Metagol$_{CF}$ cannot learn randomly chosen Context-Free languages.

Null Hypothesis 2.2 Metagol$_{CF}$ cannot outperform a state-of-the-art ILP system on learning randomly chosen Context-Free languages.

Materials and Methods Once more the randomly chosen Context-Free grammars were generated using an SLP and reduced using the Plotkin reduction algorithm (see Section 2.5). Apart from filtering grammars corresponding to finite languages, we also filtered Regular grammars which could be recognised using the pumping lemma for Context-Free grammars. However, it is not guaranteed that all Regular grammars can be filtered in this way, since it is undecidable whether a Context-Free grammar is not Regular. More specifically, if a grammar is not pumpable, then it is definitely Regular, while a pumpable grammar is not necessarily non-Regular.

The maximum number of non-terminals allowed in this experiment is 10. The examples were generated in the same way as that in the experiment about Regular-language. However, there were more duplicates in the sampled examples, because there were fewer positive examples. We used MC-TopLog with the Context-Free Meta-interpreter background knowledge against Metagol$_{CF}$.

We tested Metagol$_{CF}$'s and MC-TopLog's performance on learning Context-Free grammars using 200 different randomly chosen Context-Free grammars. The evaluation method is the same as that in Section 4.1.

Results and Discussion As shown in Figure 4b, the predictive accuracies of Metagol$_{CF}$ increase with increasing numbers of training examples. They are significantly higher than the default accuracy, therefore Null hypotheses 2.1 is refuted. Compared to Metagol$_{CF}$, MC-TopLog has consistenly higher averaged predictive accuracies. MC-TopLog average predictive accuracies surprisingly decreased after around 10 training examples. On investigation this was found to be due to the fact that larger example sets tended to contain longer examples, which increased the hypothesis space being searched. The overall effect was a radical increase in running time and simultanaeous decrease in predictive accuracy. The decrease in accuracy is consistent with the Blumer Bound [1], according to which the error bound decreases with the size of the hypothesis space.

By contrast, Metagol$_{CF}$ does a bounded search using a bottom clause so that it is feasible even though the version space is potentially infinite. Null hypothesis 2.2 is refuted with respect to running time but not predictive accuracy.

4.3 Representation Change

Null Hypothesis 3 Metagol$_{RCF}$ cannot improve performance by changing representation from Regular to Context-Free languages

Materials and Methods The material of this experiment comes from the previous two experiments. Therefore, there were 400 sampled grammars in total, half of them were Regular while the other half were mostly Context-Free and non-Regular.

We compared Metagol$_{RCF}$ (variable hypothesis space) against Metagol$_{CF}$ (fixed hypothesis space). The predictive accuracies and running time were measured as before. The results were averaged over the 400 grammars.

Results and Discussion As shown in Figure 4c, Metagol$_{RCF}$ has slightly higher predictive accuracies than Metagol$_{CF}$. This refutes Null hypothesis 3. The accuracy difference is once more consistent with the Blumer Bound [1], according to which the error bound decreases with the size of the hypothesis space.

As shown in Figure 4c, the running times of Metagol$_{CF}$ are significantly higher than Metagol$_{RCF}$. This can be explained by the fact that when the target grammar is Regular, Context-Free grammars were still considered.

5 Related work

Grammatical inference (or grammatical induction) is the process of learning a grammar from a set of examples. It is closely related to the fields of machine learning as well as the theory of formal languages and has numerous real-world applications including speech recognition (e.g. [25]), computational linguistics (e.g. [6]) and computational biology (e.g. [24]).

The problem of learning or inferring Regular languages, which can be represented by deterministic finite state automata, has been well studied and efficient automaton-based learning algorithms have existed since the 1950s [11]. Some heuristic approaches to machine learning context-free grammars [26, 10] have been investigated, though the completeness of these approaches is unclear. Although an efficient and complete approach exists for learning context-free grammars from parse trees [23], no comparable complete approach exists in the literature for learning context-free grammars from positive and negative samples of the language. According to a recent survey article learning context-free languages is widely believed to be intractable and the state of the art mainly consists of negative results [3]. There are some positive PAC (probably approximately correct) learning results concerning Regular languages (e.g. [4]), but to the best of our knowledge, these have not been extended to the context-free case. The difficulty of learning context-free languages arises from a very large search space compared to the Regular languages.

ILP, among other different learning methods, has been used for grammatical inference, for example for learning finite automata (e.g. [2]). However, as discussed in Section 1, ILP systems normally require predicate invention even for learning Regular languages. Predicate invention has been viewed as an important problem since the early days of ILP (e.g. [16]), but it is widely accepted to be a hard and under-explored topic within ILP [18].

In the Meta-interpretive Learning (MIL) framework introduced in this paper, predicate invention is done via abduction with respect to a meta-interpreter and by the introduction of first-order variables. This method is therefore related to other studies where abduction has been used for predicate invention (e.g. [9]). One important feature of MIL, which makes it distinct from other existing approaches is a bounded hypothesis space which is also ordered by θ-subsumption. We believe that this is an important advantage allowing efficient search within the space of grammars which can be very large, especially in the case of context-free languages.

6 Conclusion and further work

This paper explores the theory, implementation and experimental application of a new framework (MIL) for machine learning by adbuction with respect to a given Meta-interpreter. The MIL framework has been successfully applied to the problem of inductive inference of grammars, where our experiments indicate that it competes favourably with the state-of-the-art ILP system MC-TopLog. The MIL framework has a number of advantages with respect to the standard ILP framework. In particular, predicate invention and mutual recursion can be incorporated with ease by way of Skolem constants. The Meta-interpreter provides a natural and efficient declarative bias mechanism for controlling the search for hypotheses, which takes advantage of the efficiency and completeness of SLD resolution in Prolog. This mechanism is distinct from the use of first-order declarative bias in the form of a $\top$ theory [19, 17] since it is not assumed that the meta-interpreter entails each hypothesis.

In future work we hope to deal with a number of extensions of this study. Firstly, the experiments in this paper were all aimed at *ab initio* learning of grammars from examples, ie the case that $B_A = \emptyset$ in Definition 1. We would now like to look at the case $B_A \neq \emptyset$ in order to take account of existing bodies of background knowledge. Secondly we hope to further improve efficiency in learning Context-free grammars. One promising direction involves replacing the use of Prolog in the MIL framework with an efficient Answer Set Programming system such as Clasp [7]. Such systems use a logic programming framework with efficient constraint handling techniques to efficiently find stable models known as answer sets. ASP solvers such as Clasp compete favourably in international competitions with SAT-solvers and should be applicable to problems in the MIL framework. Thirdly we would like to extend the applications of the MIL framework to non-grammar fragments of first-order logic. In particular, we believe the framework might be fruitfully applied to the Monadic and Diadic fragments of first-order logic. Lastly, we would like to incorporate a number of other features of ILP and SRL learning systems such as probabilistic parameters (similar to SRL), noise handling and the use of compression to guide the search.

In closing we believe the MIL framework provides a promising and novel form of Inductive Logic Programming which avoids a number of the bottlenecks of existing approaches.

References

1. A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

2. H. Boström. Predicate invention and learning from positive examples only. In *10th European Conference on Machine Learning (ECML-98)*, pages 226–237. Springer, 1998.
3. C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348, 2005.
4. F. Denis. Learning regular languages from simple positive examples. *Machine Learning*, 44(1):37–66, 2001.
5. P. A. Flach and A. C. Kakas, editors. *Abductive and Inductive Reasoning.* Pure and Applied Logic. Kluwer, 2000.
6. C. Florêncio. Consistent identification in the limit of rigid grammars from strings is np-hard. *Grammatical Inference: Algorithms and Applications*, pages 729–733, 2002.
7. M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. clasp: A conflict-driven answer set solver. In Chitta Baral, Gerhard Brewka, and John Schlipf, editors, *Logic Programming and Nonmonotonic Reasoning*, volume 4483 of *Lecture Notes in Computer Science*, pages 260–265. Springer Berlin / Heidelberg, 2007.
8. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata and Formal Languages.* Addison-Wesley, Reading, MA, 1979.
9. K. Inoue, K. Furukawa, and I. Kobayashiand H. Nabeshima. Discovering rules by meta-level abduction. In L. De Raedt, editor, *Proceedings of the Nineteenth International Conference on Inductive Logic Programming (ILP09)*, pages 49–64, Berlin, 2010. Springer-Verlag. LNAI 5989.
10. P. Langley and S. Stromsten. Learning context-free grammars with a simplicity bias. In Ramon Lpez de Mntaras and Enric Plaza, editors, *Machine Learning: ECML 2000*, volume 1810 of *Lecture Notes in Computer Science*, pages 220–228. Springer Berlin / Heidelberg, 2000.
11. E.F. Moore. Gedanken-experiments on sequential machines. In C.E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 129–153. Princeton University Press, Princeton, NJ, 1956.
12. S.H. Muggleton. *Inductive Acquisition of Expert Knowledge.* Addision-Wesley, Wokingham, England, 1990.
13. S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
14. S.H. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.
15. S.H. Muggleton and C.H. Bryant. Theory completion using inverse entailment. In *Proc. of the 10th International Workshop on Inductive Logic Programming (ILP-00)*, pages 130–146, Berlin, 2000. Springer-Verlag.
16. S.H. Muggleton and W. Buntine. Machine invention of first-order predicates by inverting resolution. In *Proceedings of the 5th International Conference on Machine Learning*, pages 339–352. Kaufmann, 1988.
17. S.H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. MC-Toplog: Complete multi-clause learning guided by a top theory. 2012. Proceedings of the 21st International Conference on Inductive Logic Programming, In Press.
18. S.H. Muggleton, L. De Raedt, D. Poole, I. Bratko, P. Flach, and K. Inoue. ILP turns 20: biography and future challenges. *Machine Learning*, 86(1):3–23, 2011.
19. S.H. Muggleton, J. Santos, and A. Tamaddoni-Nezhad. TopLog: ILP using a logic program declarative bias. In *Proceedings of the International Conference on Logic Programming 2008*, LNCS 5366, pages 687–692. Springer-Verlag, 2010.
20. S.H. Muggleton N. Pahlavi. Towards efficient higher-order logic learning in a first-order datalog framework. In *Latest Advances in Inductive Logic Programming.* Imperial College Press, 2012. In Press.
21. S-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming.* Springer-Verlag, Berlin, 1997. LNAI 1228.
22. G.D. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Edinburgh University Press, Edinburgh, 1969.
23. Y. Sakakibara. Efficient learning of context-freegrammars from positive structural examples. *Information and Computation*, 97(1):23–60, 1992.
24. I. Salvador and J.M. Benedi. Rna modeling by combining stochastic context-free grammars and n-gram models. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):309–316, 2002.
25. A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201, 1995.
26. K. Vanlehn and W. Ball. A version space approach to learning context-free grammars. *Machine Learning*, 2:39–74, 1987.