Logical minimisation of meta-rules within Meta-Interpretive Learning

Andrew Cropper and Stephen H. Muggleton

Department of Computing, Imperial College London

Abstract. Meta-Interpretive Learning (MIL) is an ILP technique which uses higher-order meta-rules to support predicate invention and learning of recursive definitions. In MIL the selection of meta-rules is analogous to the choice of refinement operators in a refinement graph search. The meta-rules determine the structure of permissible rules which in turn defines the hypothesis space. On the other hand, the hypothesis space can be shown to increase rapidly in the number of meta-rules. However, methods for reducing the set of meta-rules have so far not been explored within MIL. In this paper we demonstrate that irreducible, or minimal sets of meta-rules can be found automatically by applying Plotkin's clausal theory reduction algorithm. When this approach is applied to a set of meta-rules consisting of an enumeration of all meta-rules in a given finite hypothesis language we show that in some cases as few as two meta-rules are complete and sufficient for generating all hypotheses. In our experiments we compare the effect of using a minimal set of meta-rules to randomly chosen subsets of the maximal set of meta-rules. In general the minimal set of meta-rules leads to lower runtimes and higher predictive accuracies than larger randomly selected sets of meta-rules.

1 Introduction

In [11] techniques were introduced for predicate invention and learning recursion for regular and context-free grammars based on instantiation of predicate symbols within higher-order definite clauses, or *meta-rules*. The approach was extended to a fragment of dyadic datalog in [12, 10] and shown to be applicable to problems involving learning kinship relations, robot strategies and dyadic concepts in the NELL database [4]. More recently MIL was adapted to bias reformulation when learning a hierarchy of dyadic string transformation functions for spreadsheet applications [8]. To illustrate the idea consider the meta-rule below.

Name	**Meta-Rule**
Chain	$P(x,y) \leftarrow Q(x,z), R(z,y)$

By using meta-rules as part of the proof of an example, an adapted meta-interpreter is used to build up a logic program based on a sequence of meta-substitutions into the given meta-rules which produce rules such as the following instance of the meta-rule above.

$$\text{aunt}(x,y) \leftarrow \text{sister}(x,z), parent(z,y)$$

1.1 Motivation

In [8] it was shown that within the H^2_2 hypothesis space[1] the number of programs of size n which can be built from p predicate symbols and m meta-rules is $O(m^n p^{3n})$. This result implies that the efficiency of search in MIL can be improved by reducing the number of meta-rules. In this paper we investigate ways in which minimisation of the number of meta-rules can be achieved, without loss of expressivity, by employing logical reduction techniques. This idea is illustrated in Figure 1. The two meta-rules on

[1] H^i_j consists of definite datalog logic programs with predicates of arity at most i and at most j atoms in the body of each clause.

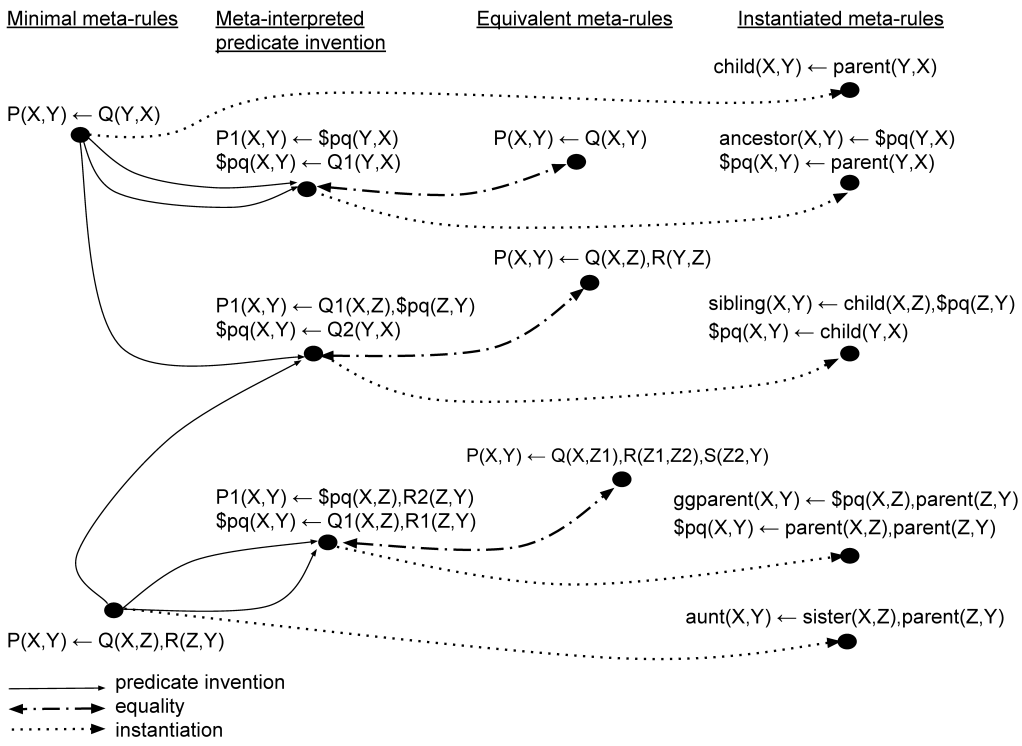


Fig. 1. The relationship between the minimal set of meta-rules, meta-rules via meta-interpreted predicate invention, and instantiated meta-rules.

the left of Figure 1 will be shown in Section 4.2 to form a minimal set for chained H_2^2. Application of such meta-rules within the meta-interpreter described in Section 5 leads to the introduction of clauses involving invented predicates such as those found in the second column of Figure 1. These definitions with invented predicates allow the search to explore the same space as if it had the meta-rules shown in the third column of Figure 1 which are found in the resolution closure of the minimal set of two meta-rules. The results in Section 4.2 show that in fact meta-rules with aribitrary numbers of literals in the clause body can be imitated in this way by the two minimal meta-rules in Figure 1.

This paper is organised as follows. Related work is discussed in Section 2. In Section 3 we describe meta-rules and the use of encapsuation. We then use encapsulation in Section 4 to show how sets of meta-rules can be reduced. This is followed in Section 5 by a description of the Prolog implementation of the Meta-Interpretretive Learning algorithm used in the experiments. In the Experiments in Section 6 we show that reduction of the meta-rules and background knowledge leads to reduced learning times with increased predictive accuracy of the learned theories. In Section 7 we conclude the paper.

2 Related work

Many approaches have been used to improve search within Inductive Logic Programming. These include probabilistic search techniques [16], the use of query packs [2] and the use of special purpose

hardware [1]. By contrast this paper explores the possibility of provably correct techniques for speeding up search based on automatically reducing the inductive meta-bias of the learner.

The use of meta-rules to build a logic program is, in some ways[2], analogous to the use of refinement operators in ILP [15, 13] to build a definite clause literal-by-literal. As with refinement operators, it seems reasonable to ask about completeness and irredundancy of any particular set of meta-rules. This question, which has not been addressed in previous papers on MIL, is investigated in Section 3. On the other hand, unlike refinement operators, meta-rules can be viewed as a form of declarative bias [6]. By comparison with other forms of declarative bias in ILP, such as modes [9, 17] or grammars [5], meta-rules are logical statements. This provides the potential for reasoning about them and manipulating them alongside normal first-order background knowledge. However, in order to do so, we need a method which supports reasoning over a mixture of higher-order and first-order clauses.

3 Meta-rules and encapsulation

In this section we introduce a program transformation technique, called *encapsulation*, which supports reasoning over meta-rules by transforming higher-order and first-order datalog statements into first-order definite clauses. We show that a higher-order datalog program has a model whenever the encapsulation of the program has a model. This allows us to show that logical reduction of encapsulated meta-rules and background knowledge can be used as a method of reducing redundancy when searching the hypothesis space. Moreover, by applying logical reduction to a complete set of meta-rules for any fragment of first-order logic we can produce a compact set of meta-rules which is complete by construction. We assume standard logic programming notation [13] throughout this section and Section 4.

3.1 Specificational and derivational meta-rules

In Section 1 we introduced the example shown below of a meta-rule M and an associated derived clause C.

Meta-Rule M	$P(x,y) \leftarrow Q(x,z), R(z,y)$
Clause C	$aunt(x,y) \leftarrow parent(x,z), sister(z,y)$

M can be viewed as a template for C. In both the meta-rule and the clause the variables x, y, z are universally quantified while in the meta-rule the variables P, Q, R are unbound (ie have no quantification). When used in the context of reasoning the unbound variables can have quantifications bound in two separate ways.

Existential quantification. Binding existential quantifiers to the unbound variables in a meta-rule produces a form of *specification* for clauses to be generated in hypotheses, so with this binding we refer to the resulting rules as *specificational meta-rules*. As in formal methods, we assume that if the specificational meta-rule M_S is the specification for clause C then $C \models M_S$.

Universal quantification. Binding universal quantifiers to the unbound variables $\mathcal{V}$ in a meta-rule produces a form appropriate for checking *derivability* for clauses in hypotheses, so with this binding we refer to the resulting rules as *derivational meta-rules*. A clause C is derivable from a derivational meta-rule M_D in the case that there exists a substitution θ with domain $\mathcal{V}$ such that $M_D\theta = C$, implying that $M_D \models C$.

[2] It should be noted that MIL uses example driven test-incorporation for finding consistent programs as opposed to the generate-and-test approach of clause refinement.

Case	Meta-rules and clause	Encapsulated clause
a)	$\$p(X,Y) \leftarrow \$q(X,Z), \$r(Z,Y)$	$m(\$p,X,Y)) \leftarrow m(\$q,X,Z)), m(\$r,Z,Y))$
b)	$P(X,Y) \leftarrow Q(X,Y)$	$m(P,X,Y) \leftarrow m(Q,X,Y)$
c)	$aunt(X,Y) \leftarrow parent(X,Z), sister(Z,Y)$	$m(aunt,X,Y)) \leftarrow m(parent,X,Z)), m(sister,Z,Y))$

Fig. 2. Three cases exemplifying encapsulation of clauses: a) Specificational meta-rule with existentially quantified variables P, Q, R by Skolem constants $\$p, \$q, \$r$, b) Derivational meta-rule with universally quantified higher-order variables P, Q, R, c) a first-order clause.

In [12, 10] specificational meta-rules were referred to simply as *meta-rules*. In this paper we distinguish between specificational and derivational meta-rules in order to clarify different forms of logical reasoning used in reduction of logical formulae involving first-order and higher-order statements.

3.2 Encapsulation

In order to support reasoning over specificational and derivational meta-rules using first-order logic we introduce a mechanism called *encapsulation*.

Definition 1. Atomic encapsulation *Let A be higher-order or first-order atom of the form $P(t_1, .., t_n)$. We say that $enc(A) = m(P, t_1, .., t_n)$ is an encapsulation of A.*

Note that the encapsulation of a higher-order atom is first-order. We now extend atomic encapsulation to logic programs.

Definition 2. Program encapsulation *The logic program $enc(P)$ is an encapsulation of the higher-order definite datalog program P in the case $enc(P)$ is formed by replacing all atoms A in P by $enc(A)$.*

Figure 2 provides examples of the encapsulation of specificational and derivational meta-rules and a first-order clause. The notion of encapsulation can readily be extended to interpretations of programs.

Definition 3. Interpretation encapsulation *Let I be a higher-order interpretation over the predicate symbols and constants in signature Σ. The encapsulated interpretation $enc(I)$ is formed by replacing each atom A in I by $enc(A)$.*

We now have the following proposition.

Proposition 1. First-order models *The higher-order datalog definite program P has a model M if and only if $enc(P)$ has the model $enc(M)$.*
Proof. *Follows trivially from the definitions of encapsulated programs and interpretations.*

We now have a method of defining entailment between higher-order datalog definite programs.

Proposition 2. Entailment *For higher-order datalog definite programs P, Q we have $P \models Q$ if and only if every model $enc(M)$ of $enc(P)$ is also a model of $enc(Q)$.*
Proof. *Follows immediately from Proposition 1.*

4 Logically reducing meta-rules

In [14] Plotkin provides the following definitions as the basis for eliminating logically redundant clauses from a first-order clausal theory.

Definition 4. Clause redundancy *The clause C is logically redundant in the clausal theory $T \wedge C$ whenever $T \models C$.*

Note that if C is redundant in $T \wedge C$ then T is logically equivalent to $T \wedge C$ since $T \models T \wedge C$ and $T \wedge C \models T$. Next Plotkin defines a reduced clausal theory as follows.

Definition 5. Reduced clausal theory *Clausal theory T is reduced in the case that it does not contain any redundant clauses.*

Plotkin uses these definitions to define a simple algorithm which given a first-order clausal theory T repeatedly identifies and removes redundant clauses until the resulting clausal theory T' is reduced.

Set of Meta-rules	**Encapsulated Meta-rules**	**Reduced set**
P(X,Y) ← Q(X,Y)	m(P,X,Y) ← m(Q,X,Y)	
P(X,Y) ← Q(Y,X)	m(P,X,Y) ← m(Q,Y,X)	m(P,X,Y) ← m(Q,Y,X)
P(X,Y) ← Q(X,Y), R(X,Y)	m(P,X,Y) ← m(Q,X,Y), m(R,X,Y)	
P(X,Y) ← Q(X,Y), R(Y,X)	m(P,X,Y) ← m(Q,X,Y), m(R,Y,X)	
P(X,Y) ← Q(X,Z), R(Y,Z)	m(P,X,Y) ← m(Q,X,Z), m(R,Y,Z)	
P(X,Y) ← Q(X,Z), R(Z,Y)	m(P,X,Y) ← m(Q,X,Z), m(R,Z,Y)	m(P,X,Y) ← m(Q,X,Z), m(R,Z,Y)
P(X,Y) ← Q(Y,X), R(X,Y)	m(P,X,Y) ← m(Q,Y,X), m(R,X,Y)	
P(X,Y) ← Q(Y,X), R(Y,X)	m(P,X,Y) ← m(Q,Y,X), m(R,Y,X)	
P(X,Y) ← Q(Y,Z), R(X,Z)	m(P,X,Y) ← m(Q,Y,Z), m(R,X,Z)	
P(X,Y) ← Q(Y,Z), R(Z,X)	m(P,X,Y) ← m(Q,Y,Z), m(R,Z,X)	
P(X,Y) ← Q(Z,X), R(Y,Z)	m(P,X,Y) ← m(Q,Z,X), m(R,Y,Z)	
P(X,Y) ← Q(Z,X), R(Z,Y)	m(P,X,Y) ← m(Q,Z,X), m(R,Z,Y)	
P(X,Y) ← Q(Z,Y), R(X,Z)	m(P,X,Y) ← m(Q,Z,Y), m(R,X,Z)	
P(X,Y) ← Q(Z,Y), R(Z,X)	m(P,X,Y) ← m(Q,Z,Y), m(R,Z,X)	

Fig. 3. Encapsulation of all 14 distinct specificational chained meta-rules from $H_m^{i=2}$ leading to a reduced set of two.

4.1 Reduction of pure dyadic meta-rules in $H_2^{i=2}$

We define $H_m^{i=2}$ as the subclass of higher-order dyadic datalog programs with lieterals of arity 2 and clause bodies consisting of at most m atoms. Now we define an associated class of chained meta-rules which will be used in the experiments.

Definition 6. Chained subset of $H_m^{i=2}$ *Let C be a meta-rule in $H_m^{i=2}$. Two literals in C are connected if and only if they share a variable or they are both connected to another literal in C. C is in the chained subset of $H_m^{i=2}$ if and only if each variable in C appears in exactly two literals and a path connects every literal in the body of C to the head of C.*

We generated a set of all chained meta-rules $H_m^{i=2}$ for this class, of which there are 14, and ran Plotkin's clausal theory reduction algorithm on this set. Figure 3 shows the 14 meta-rules together with their encapsulated form and the result of running Plotkin's algorithm which reduces the set to two meta-rules. Since, by construction this set is logically equivalent to the complete set of 14 it can be considered a *universal* set (sufficient to generate all hypotheses) for chained $H_2^{i=2}$. In the following section, we prove the same minimal set of meta-rules are complete for $H_m^{i=2}$.

4.2 Completeness theorem for $H_m^{i=2}$

We now show that two meta-rules are sufficient to generate all hypotheses in $H_m^{i=2}$.

Definition 7. *A $H_m^{i=2}$ chain rule is a meta-rule of the form $P(U_1, V) \leftarrow T_1(U_1, U_2), \ldots, T_m(U_m, V)$ where $m > 0$, P and each T_i are existentially quantified distinct predicate variables, and V and each U_i are universally quantified distinct term variables.*

We restrict ourselves to *chained* $H_m^{i=2}$. This subclass of H_m^2 is sufficiently rich for the kinship and robot examples in Section 6. We now introduce two elementary meta-rules.

Definition 8. *Let $C = P(X, Y) \leftarrow Q(Y, X)$. Then C is the inverse rule.*

Definition 9. *Let $C = P(X, Y) \leftarrow Q(X, Z), R(Z, Y)$. Then C is the $H_2^{i=2}$ chain rule.*

We now show that the inverse rule and the $H_2^{i=2}$ chain rule are sufficient to generate all hypotheses in *chained* $H_m^{i=2}$.

Lemma 1. *Let C be in $H_m^{i=2}$, L a literal in the body of C, and $m > 0$. Then resolving L with the head of the inverse rule gives the resolvent C' with the literal L' where the variables in L' are reversed.*

Proof. Trivial by construction.

Lemma 2. *Let C be the $H_m^{i=2}$ chain rule and $m > 1$. Then resolving C with the $H_2^{i=2}$ chain rule gives the resolvent R where R is the $H_{m+1}^{i=2}$ chain rule.*

Proof. Assume false. This implies that either a resolvent r of the $H_2^{i=2}$ chain rule with an $H_m^{i=2}$ chain rule does not have m+1 literals in the body or that the variables in these literals are not fully chained. Let C be the $H_2^{i=2}$ chain rule s.t. $C = P(X, Y) \leftarrow Q(X, Z), R(Z, Y)$ and let D be the $H_m^{i=2}$ chain rule s.t. $D = S(U1, V) \leftarrow T_1(U_1, U_2), T_2(U_2, U_3), \ldots, T_m(U_m, V)$. Resolving the head of C with the first body literal of D gives the unifier $\theta = \{X/U_1, Y/U_2, P/T_1\}$ and the resolvent $r = S(U_1, V) \leftarrow Q(U_1, Z), R(Z, U_2), T_2(U_2, U_3), \ldots, T_m(U_m, V)$. Since there are m-1 literals in $T_2(U_2, U_3), \ldots, T_m(U_m, V)$ then the number of literals in the body of r is $(m-1)+2 = m+1$. Thus owing to the assumption it follows that these literals are not fully chained. Since the variables in D form a chain, it follows that the variables in $T_2(U_2, U_3), \ldots, T_m(U_m, V)$ also form a chain, so all variables in r form a chain. This contradicts the assumption and completes the proof.

Lemma 3. *Let C_1 be the inverse rule, C_2 the $H_m^{i=2}$ chain rule, S_m the set of meta-rules in $H_m^{i=2}$ with exactly m literals in the body, and $m > 0$. Then $\{C_1, C_2\} \models R$ for every R in S_m.*

Proof. R can differ from C_2 in only two ways: (1) the order of the variables in a literal and (2) the order of the literals in the body. Case 1. By lemma 1 we can resolve any literal L in R with C_1 to derive R' with the literal L' such that the variables in L' are reversed. Case 2. Let $C_2 = P \leftarrow \ldots, T_i, T_j, \ldots, T_m$ and R $= P \leftarrow, \ldots, T_j, T_i \ldots, T_m$ where $T_1, \ldots, T_m$ are literals and the predicate symbols for T_i and T_j are existentially quantified variables. Then we can derive R from C_2 with the substitution $\theta = \{T_i/T_j, T_j/T_i\}$. Since these two cases are exhaustive, this completes the proof.

Theorem 1. *Let C_1 be the inverse rule, C_2 the $H_2^{i=2}$ chain rule, S_m the set of meta-rules in $H_m^{i=2}$, and $m > 0$. Then $\{C_1, C_2\} \models R$ for every R in S_m.*

Proof. By induction on m.

1. Suppose $m = 1$. Then $S_1 = \{(P(X,Y) \leftarrow Q(X,Y)), (P(X,Y) \leftarrow Q(Y,X))\}$. By lemma 1, we can resolve C_1 with $C_1{}'$ to derive $C_3 = P(X,Y) \leftarrow Q(X,Y)$. The set $\{C_1, C_3\} = S_1$, thus the proposition is true for $m = 1$.
2. Suppose the theorem holds if $k \leq m$. By lemma 2 we can resolve the $H_2^{i=2}$ chain rule with the $H_m^{i=2}$ chain rule to derive the $H_{m+1}^{i=2}$ chain rule. By the induction hypothesis there is a $H_m^{i=2}$ chain rule, so we can resolve this with C_2 to derive the $H_{m+1}^{i=2}$ chain rule. By lemma 3, we can derive any R in $H_m^{i=2}$ with exactly m literals, thus completing the proof.

4.3 Representing $H_m^{i=2}$ programs in $H_2^{i=2}$

We now show that $H_2^{i=2}$ is a normal form for $H_m^{i=2}$.

Theorem 2. *Let C be a meta-rule in $H_m^{i=2}$ and $m > 2$. Then there is an equivalent theory in $H_2^{i=2}$.*

Proof. We prove by construction. Let C be of the form $P \leftarrow T_1, \ldots, T_m$. For any literal T_i in the body of C of the form $T(U_{i+1}, U_i)$ introduce a new predicate symbol $\$s_i$, a new clause $(\$s_i(X,Y) \leftarrow T_i(Y,X))$, and then replace $T_i(U_{i+1},\ U_i)$ in C with $\$s_i(U_i, U_{i+1})$. Step 2. Introduce new predicates symbols $(\$p_1, \ldots, \$p_{m-2})$ and new clauses $(P(X,Y) \leftarrow T_1(X,Z), \$p_1(Z,Y)), (\$p_1(X,Y) \leftarrow T_2(X,Z), \$p_2(Z,Y))$, $\ldots, (\$p_{m-2}(X,Y) \leftarrow t_{m-1}(X,Z), t_m(Z,Y))$. Step 3. Remove the original clause C from the theory. You now have an equivalent theory in $H_2^{i=2}$.

This theorem is exemplified below.

Example 1. Let $C = P(U_1,V) \leftarrow T_1(U_1,U_2), T_2(U_3,U_2), T_3(U_3,U_4), T_4(U_4,V)$. Notice that literal T_2 is in non-standard form, i.e. the variables in T_2 are in the order (U_{i+1}, U_i) and not (U_i, U_{i+1}).We now proceed to a theory in $H_2^{i=2}$ equivalent to C. Step 1. Introduce a new predicate symbol $\$s_2$, a new clause $\$s_2(X,Y) \leftarrow T_3(Y,X)$, and replace T_3 in C with $\$s_2(U_2,U_3)$ so that $C = P(U_1,V) \leftarrow T_1(U_1,U_2), \$s_2(U_2,U_3), T_3(U_3,U_4), T_4(U_4,V)$. Step 2. Introduce new predicate symbols $\$p_1$ and $\$p_2$ and the following new clauses:

$$P(X,Y) \leftarrow T_1(X,Z), \$p_1(Z,Y)$$
$$\$p_1(X,Y) \leftarrow \$s_2(X,Z), \$p_2(Z,Y)$$
$$\$p_2(X,Y) \leftarrow T_3(X,Z), T_4(Z,Y)$$

Step 3. After removing C you are left with the following theory, which is equivalent to C.

$$\$s_2(X,Y) \leftarrow T_2(Y,X)$$
$$P(X,Y) \leftarrow T_1(X,Z), \$p_1(Z,Y)$$
$$\$p_1(X,Y) \leftarrow \$s_2(X,Z), \$p_2(Z,Y)$$
$$\$p_2(X,Y) \leftarrow T_3(X,Z), T_4(Z,Y)$$

5 Implementation

Figure 4 shows the implementation of MetagolD used in the experiments in Section 6. In this implementation the generalised Meta-Interpreter (Figure 4a) has a similar form to a standard Prolog meta-interpreter. The differences are as follows. While Prolog program represents its program by a set of first-order definite clauses, Metagol represents its program by a set of meta-substitutions which yield first-order clauses when applied to the associated meta-rules. When a Prolog meta-interpreter repreatedly unifies the first atom in the goal with the head of a clause selected from the clause base it updates the bindings in the answer set. By contrast, when the meta-interpretrive learner in Figure 4a unifies *Atom* in the goal with the head of the meta-rule *Atom :- Body* from the meta-rule base the *abduce* predicate adds the meta-substitution *MetaSub* to the set of meta-substitutions comprising the hypothesised program[3]. Figure 4b) shows the set of dyadic meta-rules used within the experiments.

6 Experiments

These experiments compare the effect of using a minimal set of meta-rules to a maximum set of meta-rules. We also consider the intermediate cases, i.e. randomly chosen subsets of the maximum set of meta-rules. We define Metagol$_{min}$ and Metagol$_{max}$ as variants of Metagol$_D$ [12, 10] which use the minimum and maximum sets of meta-rules respectively,

6.1 Experimental hypotheses

The following null hypotheses were tested:

Null Hypothesis 1. Metagol$_{min}$ cannot achieve higher predictive accuracy than Metagol$_{max}$.
Null Hypothesis 2. Metagol$_{min}$ and Metagol$_{max}$ have the same learning times.

6.2 Learning Kinship Relations

We used the kinship dataset from [7][4], which contains 12 dyadic relations: *aunt, brother, daughter, father, husband, mother, nephew, niece, sister, son,* and *wife*, and 104 examples.

[3] The *OrderTest* represents a meta-rule associated constraint which ensures termination, as explained in [12, 10].
[4] https://archive.ics.uci.edu/ml/datasets/Kinship

a) Generalised meta-interpreter

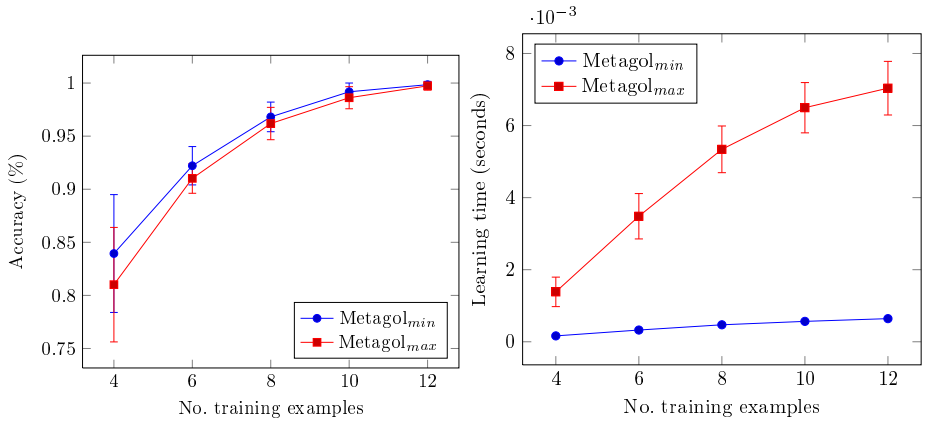
```
prove([],Prog,Prog).
prove([Atom|As],Prog1,Prog2) :-
        metarule(RuleName,MetaSub,(Atom :- Body),OrderTest),
        OrderTest,
        abduce(metasub(RuleName,MetaSub),Prog1,Prog3),
        prove(Body,Prog3,Prog4),
        prove(As,Prog4,Prog2).
```

b) Meta-rules for dyadic fragment

```
metarule(inverse,[P,Q],([P,X,Y] :- [[Q,Y,X]]),
        (pred_above(P,Q),obj_above(X,Y))).
metarule(chain,[P,Q,R],([P,X,Y] :- [[Q,X,Z],[R,Z,Y]]),
        (pred_above(P,R),obj_above(X,Z),obj_above(Z,Y))).
```

Fig. 4. MetagolD Prolog representation of a) Generalised meta-interpreter and b) Dyadic fragment meta-rules

Experiment 1: Metagol$_{min}$ vs Metagol$_{max}$. We randomly selected n examples, half positive and half negative, of each target kinship relation and performed leave-one-out-cross-validation. All relations excluding the target relation were used as background knowledge. Predictive accuracies and associated learning times were averaged over each relation over 200 trials. Fig.5 shows that Metagol$_{min}$ outperforms Metagol$_{max}$ in predictive accuracy. A McNemar's test at the 0.001 level confirms the significance, rejecting null hypothesis 1. This can be explained by the larger hypothesis space searched by Metagol$_{max}$ and the Blumer bound [3], which says that a larger search space leads to higher predictive error. Fig.5 shows that Metagol$_{min}$ outperforms Metagol$_{max}$ in learning time with Metagol$_{max}$ taking up to seven times longer. Thus null hypothesis 2 is clearly rejected.

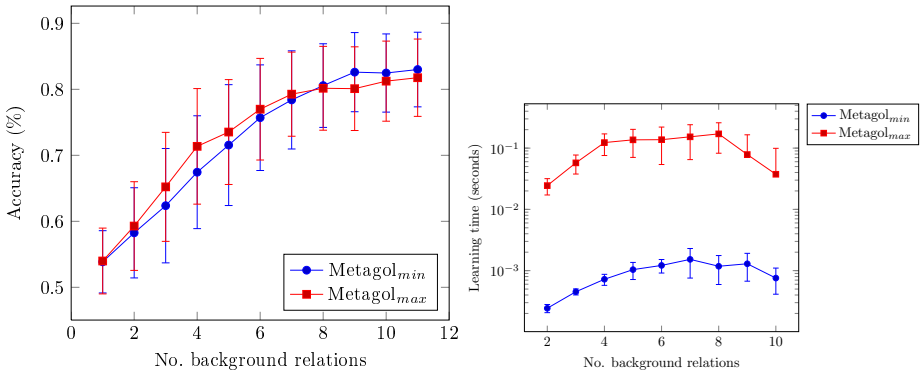


(a) Predictive Accuracies (b) Learning Times

Fig. 5. Performance of full enumeration (Metagol$_{max}$) and logically reduced (Metagol$_{min}$) sets of metarules when varying number of training examples.

Experiment 2: Sampled background relations. To explore the effect of partial background information we repeated the same steps as in experiment 1 but randomly selected p relations as the background knowledge. Fig.6(a) shows that Metagol$_{max}$ outperforms Metagol$_{min}$ in predictive accuracy when n=4, but when the number of background relations approaches eight Metagol$_{min}$ starts to outperform Metagol$_{max}$. This is because Metagol$_{min}$ has to derive meta-rules (e.g. the identity rule) through predicate invention to replace missing relations and is reaching the logarithmic depth bound on the iterative deepening search. A subsequent experiment performed without a logarithmic depth bound confirmed this. Fig.6(b) shows that Metagol$_{max}$ has considerably longer learning time compared to Metagol$_{min}$. We repeated this experiment for n>4 but as the number relations approaches six the learning time of Metagol$_{max}$ became prohibitively slow, whereas Metagol$_{min}$ was still able to learn definitions (results omitted for brevity).

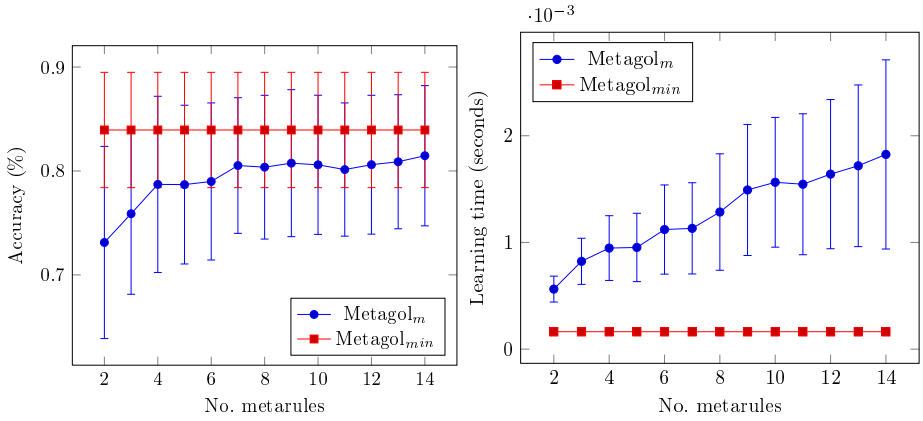
Experiment 3: Sampled meta-rules. For this experiment we considered a version of Metagol supplied with a subset of m meta-rules randomly selected from the set of all meta-rules and repeated the same steps as experiment 1. Figures 7(a) and 7(b) show the predictive accuracies and associated



(a) Predictive Accuracies (b) Learning Times

Fig. 6. Performance of full enumeration (Metagol$_{max}$) and logically reduced (Metagol$_{min}$) sets of metarules when varying number of background relations.

learning times when n=4 where Metagol$_m$ represents Metagol loaded with m sampled meta-rules. The corresponding results for Metagol$_{min}$ from experiment 1 are provided for comparison. These results show that Metagol$_{min}$ outperforms Metagol$_m$ in predictive accuracy and learning time. The results for other values of n (omitted for brevity) were consistent with when n=4.



(a) Predictive Accuracies (b) Learning Times

Fig. 7. Peformance when sampling metarules from the full enumeration.

6.3 Learning Robot Strategies

Imagine a robot in a 2-dimensional space which can perform six dyadic actions: *move_left, move_right, move_forwards, move_backwards, grab_ball,* and *drop_ball*. The robot's state is represented as a list with three elements [RobotPos,BallPos,HasBall], where RobotPos and BallPos are coordinates and HasBall

is a boolean representing whether the robot has the ball. The robot's task is to move the ball to a destination. Suppose we have the following example:

$$move\_ball([0/0, 0/0, false], [2/2, 2/2, false])$$

Here the robot and ball both start at (0,0) and both end at (2,2). Given this example, Metagol$_{min}$ learns the following strategy:

s2(X,Y)← move_forwards(X,Z),drop_ball(Z,Y)
s2(X,Y)← grab_ball(X,Z),move_right(Z,Y)
s3(X,Y)← s2(X,Z), s2(Z,Y)
move_ball(X,Y)← s3(X,Z),s3(Z,Y)

Here the robot grabs the ball, moves right, moves forward, drops the ball, and then repeats this process. Metagol$_{min}$ also learns an alternative strategy where the robot performs grab_ball and drop_ball only once. Both are considered equal because both are the same length. If we wanted to prefer solutions which minimise the number of grabs and drops we would need to associate costs with different operations. Now suppose that we exclude move_right from the background knowledge, given the original example, Metagol$_{min}$ learns the strategy:

s2(X,Y)← move_backwards(X,Z),drop_ball(Z,Y)
s2(X,Y)← grab_ball(X,Z),move_left(Z,Y)
s3(X,Y)← s2(X,Z),s2(Z,Y)
s4(X,Y)← s3(X,Z),s3(Z,Y)
move_ball(X,Y)← s4(Y,X)

Metagol$_{min}$ found this solution by inverting the high-level action *s4*, thus allowing the actions *move_left*, *move_backwards*, and *grab_ball* to indicate their respective inverse actions *move_right*, *move_forwards*, and *drop_ball*. If we also remove move_forwards and drop_ball from the background knowledge, Metagol$_{min}$ learns a strategy which uses only five clauses and three primitives, compared to the original four clause solution which used six primitives. The construction of an inverse plan is familiar to retrograde analysis of positions in chess, in which you go backwards from an end position to work out the moves necessary to get there from a given starting position. We compared Metagol$_{min}$ and Metagol$_{max}$ for learning robot strategies, but the learning time of Metagol$_{max}$ was prohibitively slow in all but trivial circumstances, whereas Metagol$_{min}$ was able to find optimal solutions with acceptable learning times.

7 Conclusion and future work

We have shown that just two meta-rules are complete and sufficient for generating all hypotheses in the case of $H_2^{i=2}$. However, this set of reduced meta-rules is insufficient for generating all hypotheses in the case of H_2^2. For example, there is no way to generate any hypothesis containing monadic predicates. Future work should address this.

Our experiments suggest that the minimal set of meta-rules achieves higher predictive accuracies and lower learning times than the maximum set. In Section 6.2 we explored the intermediate cases by sampling meta-rules from the maximum set. The results suggested that there was no benefit in using more meta-rules than the minimum set. However, this is not always the case. For example, the dyadic string transformation functions described in [8] require only the chain meta-rule. Thus the optimal set

of meta-rules for this task is clearly a subset of the reduced minimal sets described in this paper. One future direction is to investigate learning the optimal set of meta-rules as to minimise the hypothesis space. One idea is to start with a single meta-rule and to invent new meta-rules when required. This would require a MIL equivalent of refinement operators and the need to develop a theory regarding subsumption order between meta-rules. Closely related to this is learning orderings of meta-rules. For example, when learning robot strategies in Section 6.3 the chain rule was used more often than the inverse rule. It would be preferable to learn an ordering for the meta-rules to prevent the search from entering unnecessary branches.

7.1 Future work

Results in this paper have been developed for the case of finding minimal sets of meta-rules for the chained $H_m^{i=2}$ class of definite clause programs. In future work we intend to see whether the same approach can be extended to broader classes of meta-rules.

In addition, the ability to encapsulate background knowledge, as demonstrated in Figure 2 indicates that it might be possible to minimise the meta-rules together with a given set of background clauses. Preliminary experiments seem to indicate that this is possible, and we would aim to report results on this approach in any extended version of this paper.

Finally, when learning robot strategies in Section 6.3 we observed a situation when Metagol learned hypotheses of equal length, but one was preferable to the other. This suggests future work investigating a version of MIL which associates costs with different operations, so that Metagol prefers solutions with lower costs. This would have applications in robot strategy learning and path planning problems.

Acknowledgements

The first author acknowledges the support of the BBSRC and Syngenta in funding his PhD Case studentship. The second author would like to thank the Royal Academy of Engineering and Syngenta for funding his present 5 year Research Chair.

References

1. S.H. Muggleton A. Fidjeland, W. Luk. Scalable acceleration of inductive logic programs. In *IEEE international conference on field-programmable technology*, pages 252 – 259. IEEE, 2002.
2. Hendrik Blockeel, Luc Dehaspe, Bart Demoen, Gerda Janssens, Jan Ramon, and Henk Vandecasteele. Improving the efficiency of inductive logic programming through the use of query packs. *Journal of Artificial Intelligence Research*, 16(1):135–166, 2002.
3. 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.
4. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*, 2010.
5. W. Cohen. Grammatically biased learning: Learning logic programs using an explicit antecedent description language. *Artificial Intelligence*, 68:303–366, 1994.
6. L. De Raedt. Declarative modeling for machine learning and data mining. In *Proceedings of the International Conference on Algorithmic Learning Theory*, page 12, 2012.
7. G E Hinton. Learning distributed representations of concepts. *Artificial Intelligence*, 40:1–12, 1986.
8. D. Lin, E. Dechter, K. Ellis, J.B. Tenenbaum, and S.H. Muggleton. Bias reformulation for one-shot function induction. In *Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI 2014)*, pages 525–530, Amsterdam, 2014. IOS Press.
9. S.H. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

10. S.H. Muggleton and D. Lin. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In *Proceedings of the 23rd International Joint Conference Artificial Intelligence (IJCAI 2013)*, pages 1551–1557, 2013.
11. S.H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94:25–49, 2014.
12. S.H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning*, 2015. To appear.
13. S-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag, Berlin, 1997. LNAI 1228.
14. G.D. Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburgh University, August 1971.
15. E.Y. Shapiro. *Algorithmic program debugging*. MIT Press, 1983.
16. A. Srinivasan. A study of two probabilistic methods for searching large spaces with ilp. Technical Report PRG-TR-16-00, Oxford University Computing Laboratory, Oxford, 2000.
17. A. Srinivasan. The ALEPH manual. *Machine Learning at the Computing Laboratory, Oxford University*, 2001.