Efficiently Encoding Meta-Interpretive Learning by Answer Set Programming (*Work in Progress*)*

Tobias Kaminski[1], Thomas Eiter[1], and Katsumi Inoue[2]

[1] Technical University of Vienna (TU Wien), Vienna, Austria
{kaminski,eiter}@kr.tuwien.ac.at
[2] National Institute of Informatics, Tokyo, Japan
inoue@nii.ac.jp

Abstract. Meta-Interpretive Learning (MIL) learns logic programs from examples by instantiating meta-rules, which is implemented by the Metagol system using Prolog. Based on previous work wrt. solving MIL by employing Answer Set Programming, in this work-in-progress paper we describe a modification of a previous MIL-encoding which prunes the search space more effectively by simulating a top-down search as performed by Prolog. Initial experiments show that our new encoding can significantly speed up the inductive learning process.

1 Introduction

Meta-Interpretive Learning (*MIL*) [4] is a recent approach for learning logic programs from positive and negative examples wrt. background knowledge by instantiating so-called *meta-rules*, which specify the shapes of rules that may be used in the induced program. The formalism is very powerful as it enables *predicate invention* and supports learning of recursive programs. The *Metagol* system is an efficient implementation of MIL based on a classical *Prolog* meta-interpreter, which uses a query-driven procedure to effectively steer the instantiation of meta-rules needed for deriving positive examples. Yet, the sequential treatment of examples that goes along with it can also be disadvantageous since the derivability of negative examples is only detected after all positive examples are covered. This has been identified as a bottleneck of Metagol [2].

Viewing MIL as combinatorial search problem, it can alternatively be encoded by *Answer Set Programming* (*ASP*), which may speed up solving significantly due to early detection and learning from conflicts arising due to the derivability of negative examples. Muggleton et al. already observed that ASP can have an advantage for MIL over Prolog due to effective pruning [3], but their encoding only used one specific meta-rule and was tailored to inducing grammars. However, when multiple meta-rules are used and the background knowledge is more extensive, a straightforward encoding easily yields a huge search space that makes solving infeasible. For this reason, more sophisticated MIL-encodings have been proposed [2]. They limit the search space by interleaving derivations at the object level and the meta level such that only potentially

* This research has been supported by the Austrian Science Fund (FWF) projects P27730 and W1255-N23.

relevant instantiations of meta-rules wrt. pieces of information are generated which are already derivable in a *bottom-up* fashion.

The approach by Kaminski et al. employs HEX-programs [1], which extend ASP with a bidirectional information exchange between a program and arbitrary external sources of computation via special *external atoms*. Their basic encoding utilizes external atoms to outsource background knowledge that defines manipulations of complex terms such as lists and strings, which is easy to realize in Prolog but less supported by ASP. Moreover, Kaminski et al. identified a class of MIL-problems widely studied which can be solved efficiently by their HEX-encodings, and showed empirically that the performance can be increased compared to Metagol by employing ASP.

Even though conflicts resulting from negative examples can be propagated effectively in the approach of Kaminski et al., the query-driven procedure of Metagol still has an advantage wrt. finding derivations of positive examples as it generates exactly one instantiation of a meta-rule for proving the next subgoal during SLD-resolution. In comparison, the hypothesis space generated by the HEX-encodings is larger and potentially generates many rules that are not required for deriving any subgoal due to its bottom-up nature, such that there is still room for improvement. Ideally, an approach would combine the effective *top-down* derivation of positive examples of Metagol with the propagation of conflicts resulting from negative examples for early backtracking. However, the obstacle wrt. this combination so far consisted in the fact that a naive top-down encoding in ASP that models the original search space of Metagol requires the generation of all ground instances of instantiated meta-rules due to the *grounding bottleneck* of ASP, which is highly infeasible.

In this paper, we present work in progress towards solving the mentioned challenge regarding a simulation of backward-chaining in ASP. Our new MIL-encoding works in two stages, by first constructing all ground instances of instantiated meta-rules that are possibly relevant for deriving positive examples in a bottom-up manner as before, and simulating a top-down search for a hypothesis that models all positive examples only wrt. these rule instances. This way, the size of the grounding can be restricted drastically, and first experiments using our new encoding indicate that simulating a top-down search can further increase the performance of solving MIL-problems by ASP.

In the next section, we first recall the essential notions of MIL and demonstrate the bottom-up MIL-encoding of Kaminski et al. by means of an example. In Section 3, we describe a modified encoding that simulates the top-down derivation of positive examples and present preliminary empirical results, before we conclude and shortly discuss ongoing work in Section 4.

2 Previous Bottom-Up MIL-Encoding

In this section, we summarize the main aspects of the forward-chained HEX-MIL-encoding by Kaminski et al. [2] for solving MIL-problems.

Formally, a *MIL-problem* in our setting is a quadruple $\mathscr{M} = (B, E^+, E^-, \mathscr{R})$, where B is the *background knowledge* represented by a logic program, E^+ and E^- are finite sets of binary ground atoms called *positive* resp. *negative examples*, and $\mathscr{R}$ is a finite set of meta-rules. At this, we consider meta-rules of the form $P(x,y) \leftarrow Q_1(x_1,y_1),...,$

$Q_k(x_k,y_k), R_1(z_1),...,R_n(z_n)$, where P, Q_i, $1 \leq i \leq k$, and R_j, $1 \leq j \leq n$, are higher-order variables, and x,y,x_i,y_i, $1 \leq i \leq k$, and z_j, $1 \leq j \leq n$, are first-order variables s.t. x and y also occur in the body. A *meta-substitution* of a meta-rule R is an instantiation of R where all higher-order variables are substituted by predicate symbols. A *solution* for $\mathscr{M}$ is a *hypothesis* $\mathscr{H}$ consisting of a set of meta-substitutions of meta-rules in $\mathscr{R}$ s.t. $B \cup \mathscr{H} \models e^+$ for each $e^+ \in E^+$ and $B \cup \mathscr{H} \not\models e^-$ for each $e^- \in E^-$.

The forward-chained HEX-MIL-encoding of Kaminski et al. is restricted to MIL-problems where $\mathscr{R}$ only contains *forward-chained* meta-rules, i.e. meta-rules where the elements x,y in the binary head $p(x,y)$ are connected via a path $p_1(x_1,x_2)$, $p_2(x_2,x_3)$, $\ldots$, $p_k(x_k,x_{k+1})$ in the body, where $x = x_1$ and $x_{k+1} = y$.

Example 1. Consider a MIL-problem $\mathscr{M}$ with $\mathscr{R} = \{P(x,y) \leftarrow Q(x,z),R(z,y); P(x,y) \leftarrow Q(x,y),R(y)\}$ and $B = \{remove([X|R],R) \leftarrow; empty([]) \leftarrow\}$, where the predicate *remove* drops the first element from a list and *empty* checks if a list is empty. Furthermore, let $E^+ = \{p([a,a],[]),p([b,b],[])\}$, $E^- = \{p([a,a,a],[a]),p([b,b,b],[])\}$ be the given positive and negative examples. Accordingly, a corresponding hypothesis intuitively needs to remove all elements (only) from a list containing two elements provided as the first argument of an example to yield an empty list in the second argument position. This is, e.g., captured by the hypothesis $\mathscr{H} = \{p(x,y) \leftarrow remove(x,z), p1(z,y); p1(x,y) \leftarrow remove(x,y),empty(y)\}$, where $p1$ is an invented predicate.

The MIL-encodings of Kaminski et al. utilize *external atoms* in rule bodies, provided by the HEX-extension [1] of ASP, for interfacing the respective background knowledge. At this, input values in form of predicates and constants are provided to an external source which computes corresponding output values. For instance, if the background knowledge defines a predicate that drops the first element of a list provided as first argument and returns the resulting list as second argument, an external atom $\&remove[X](Y)$ might be used, which would evaluate to true for all instantiations of X and Y where X is a Prolog list and Y is obtained from X by removing its first element.

As in the Metagol implementation of MIL, we assume a given set $\mathscr{S}$ of *Skolem predicates* that can be used for predicate invention, and an ordering $\succeq_{\mathscr{P}}$ over all predicates that can be instantiated for higher-order variables in meta-rules, which is used for constraining the set of permissible meta-substitutions. Together with facts $ord(p,q)$ for all predicates p and q s.t. $p \succeq_{\mathscr{P}} q$ and facts $pos\_ex(a,b)$ and $neg\_ex(a,b)$ for all given positive and negative examples, resp., the rules (B1)-(B12) in Figure 1 correspond to the *forward-chained* HEX-*MIL-encoding* for $\mathscr{M}$ from Example 1 as introduced by Kaminski et al., which guesses new meta-substitutions based on facts that are already derivable in a bottom-up manner.

The rules (B1) and (B2) import all terms that can be derived from unary and binary predicates defined by the background knowledge wrt. already imported terms in an inductive manner via rule (B5) by utilizing external atoms. At this, the import starts from terms that occur as the first argument of an example according to rules (B3) and (B4), and new terms are added incrementally to the extension of the predicate *state*. The predicate *meta* contains all meta-substitutions added to an induced hypothesis. The rules (B6)-(B8) define the predicate *ded*, which captures all atoms that can be deduced from the meta-substitutions in a guessed hypothesis together with the imported background

$$binary\_bg(remove,X,Y) \leftarrow \&remove[X](Y), state(X). \quad \text{(B1)}$$
$$unary\_bg(empty,X) \leftarrow \&empty[X](), state(X). \quad \text{(B2)}$$
$$state(X) \leftarrow pos\_ex(\_,X,\_). \quad \text{(B3)}$$
$$state(X) \leftarrow neg\_ex(\_,X,\_). \quad \text{(B4)}$$
$$state(Y) \leftarrow binary\_bg(\_,\_,Y). \quad \text{(B5)}$$
$$ded(P,X,Y) \leftarrow binary\_bg(P,X,Y). \quad \text{(B6)}$$
$$ded(P,X,Y) \leftarrow meta(postcon,P,Q,R), ded(Q,X,Y), unary\_bg(R,Y). \quad \text{(B7)}$$
$$ded(P,X,Y) \leftarrow meta(chain,P,Q,R), ded(Q,X,Z), ded(R,Z,Y). \quad \text{(B8)}$$
$$\leftarrow neg\_ex(P,X,Y), ded(P,X,Y). \quad \text{(B9)}$$
$$\{meta(chain,P,Q,R)\} \leftarrow ord(P,Q), ord(P,R), ded(Q,X,Z), ded(R,Z,Y). \quad \text{(B10)}$$
$$\{meta(postcon,P,Q,R)\} \leftarrow ord(P,Q), ded(Q,X,Y), unary\_bg(R,Y). \quad \text{(B11)}$$
$$\leftarrow pos\_ex(P,X,Y), \mathrm{not}\, ded(P,X,Y). \quad \text{(B12)}$$

Fig. 1. Bottom-up MIL-encoding of $\mathscr{M}$ from Example 1.

knowledge. In turn, new meta-substitutions are guessed by rules (B10) and (B11), where (B10) generates substitutions of the first (*chain*) meta-rule in $\mathscr{R}$, and (B11) of the second (*postcon*) meta-rule; and the heads of the rules encode that an arbitrary number of instances of the head may be true whenever the rule body is satisfied. Importantly, only those meta-substitutions can be added where the body atoms of some ground instance can already be derived in a bottom-up manner, i.e. only if it is potentially useful for deriving a positive example. Accordingly, the encoding interleaves bottom-up derivations at the object level and guesses at the meta level; this constitutes a main difference to the first ASP-encoding of MIL by Muggleton et al. [3]. Finally, the constraint in rule (B12) ensures that all positive examples can be derived. Solutions of $\mathscr{M}$ correspond to the sets of atoms with predicate *meta* contained in the answer sets of the encoding.

3 New Top-Down MIL-Encoding

As already mentioned in Section 1, the MIL-encoding in Figure 2 merely guesses meta-substitutions for deriving positive examples based on pieces of information that are derived in a bottom-up fashion, such that the constraint in rule (B12) is required to ensure that all positive examples are indeed entailed by the guessed hypothesis. This has the disadvantages that (i) missing rules for deriving some subgoal in the derivation of positive examples are only detected late after checking the constraint, and (ii) more meta-substitutions than necessary may be added to a solution. The latter increases the size of the obtained solution and makes the derivability of negative examples more likely. Accordingly, in this section, we introduce a novel variant of our previous forward-chained HEX-MIL-encoding, which selects exactly one ground instance of a meta-substitution for each subgoal that needs to be derived in the derivation of positive examples; in a way, it simulates backward-chaining as performed by Prolog.

$$ded\_a(bg,P,n,n,X,Y,n) \leftarrow binary\_bg(P,X,Y). \quad \text{(T10)}$$

$$ded\_a(postcon,P,Q,R,X,Y,n) \leftarrow ord(P,Q), ded\_a(\_,Q,\_,\_,X,Y,\_), unary\_bg(R,Y). \quad \text{(T11)}$$

$$ded\_a(chain,P,Q,R,X,Y,Z) \leftarrow ord(P,Q), ord(P,R), ded\_a(\_,Q,\_,\_,X,Z,\_), ded\_a(\_,R,\_,\_,Z,Y,\_). \quad \text{(T12)}$$

$$goal(P,X,Y) \leftarrow pos\_ex(P1,X,Y). \quad \text{(T13)}$$

$$goal(Q,X,Z) \leftarrow ded\_u(chain,P,Q,R,X,Y,Z). \quad \text{(T14)}$$

$$goal(R,Z,Y) \leftarrow ded\_u(chain,P,Q,R,X,Y,Z). \quad \text{(T15)}$$

$$goal(Q,X,Y) \leftarrow ded\_u(postcon,P,Q,R,X,Y,\_). \quad \text{(T16)}$$

$$\{ded\_u(M,P,Q,R,X,Y,Z) : ded\_a(M,P,Q,R,X,Y,Z)\} = 1 \leftarrow goal(P1,X,Y). \quad \text{(T17)}$$

$$meta(M,P,Q,R) \leftarrow ded\_u(M,P,Q,R,X,Y,Z), M \neq bg. \quad \text{(T18)}$$

Fig. 2. Top-down MIL-encoding of $\mathscr{M}$ from Example 1.

Now, a straightforward approach would generate all possible instances of meta-substitutions during grounding and select one of them for each subgoal. This would make grounding infeasible. Hence, we need to limit the number of ground meta-substitutions produced during grounding. To this end, we first compute all relevant instances that can be obtained from the imported background knowledge and meta-rules in a bottom-up fashion, similar as done by the rules (B7) and (B8) of the previous encoding. The difference is that meta-substitutions are not guessed but all possible meta-substitutions are taken into account. Accordingly, an envelope for the ground instances of meta-substitutions that are used for the top-down derivation of positive examples is generated first.

Our new top-down MIL-encoding for $\mathscr{M}$ from Example 1 is obtained from the encoding presented in Section 2 by using rules (B1)-(B9) in Figure 1 together with the new rules (T10)-(T18) in Figure 2. The crucial difference consists in the fact that meta-substitutions which are contained in an induced hypothesis are not guessed as by rules (B10) and (B11) in the previous encoding, but added via rule (T18) when some respective ground instance is selected in a recursive derivation of subgoals by rules (T13)-(T17). As a result, the constraint (B12) can also be omitted since it is already ensured that each positive example is entailed by the resulting hypothesis. First, the rules (T10)-(T12) produce *all* ground instances of meta-subsitutions that can be derived, starting from the background knowledge using all possible meta-substitutions, and store them in the extension of the predicate *ded_a*. Second, to simulate the top-down search for proving the positive examples, rule (T13) defines positive examples as initial goals. Rule (T17) states that for each new subgoal there needs to be exactly one ground instance of a meta-substitution that allows to derive it, where *used* instances are stored in the extension of the predicate *ded_u*. The rules (T14)-(T16) recursively add new subgoals to the predicate *goal*, based on the rule instances selected by rule (T17). Finally, rule (T18) accumulates all meta-substitutions representing a computed hypothesis in the extension of the predicate *meta*.

We have tested the new top-down MIL-encoding utilizing two benchmark problems already used by Kaminski et al. [2], and compared it to Metagol as well as the previous

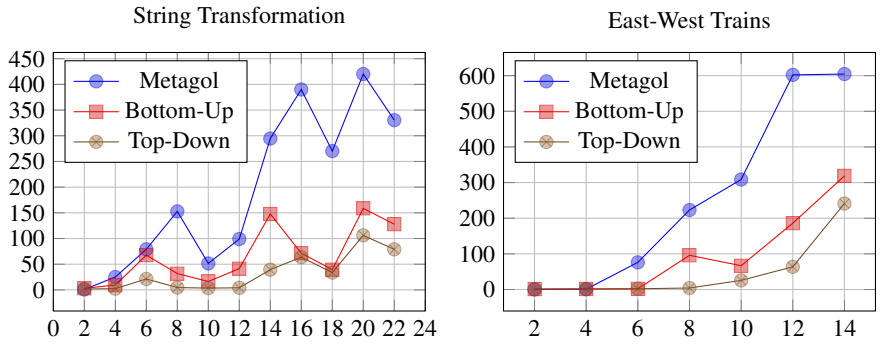

Fig. 3. Benchmark results of comparing a top-down and a bottom-up ASP-encoding to Metagol.

bottom-up MIL-encoding by Kaminski et al. Figure 3 shows the corresponding results, where the x-axis indicates the problem size and the y-axis shows the running time in seconds (averaged over 20 instances in case of the *string transformation* benchmark, and over 10 instances for the *east-west trains* benchmark).[1] The timeout was set to 600 seconds for both experiments. We found that by using our new encoding, an additional speed up over Metagol and wrt. the previous MIL-encoding is possible.

4 Conclusion and Ongoing Work

In this work-in-progress paper, we have reported initial results regarding a novel ASP-encoding for solving MIL-problems, which modifies a previously developed encoding by simulating backward-chaining. The next steps of our work consist in defining a general formalization of the new encoding applicable to arbitrary forward-chained MIL-problems, and to show its soundness and completeness. Moreover, we are planning to investigate the relations of our encodings to *magic sets* in ASP, which also serve the purpose of combining advantages of bottom-up and top-down evaluation in logic programming. Finally, further goals of future work are to also apply our new techniques to the second MIL-encoding of Kaminski et al. [2] that abstracts from object-level terms, and to perform more extensive tests using additional benchmark problems.

References

1. Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C., Schüller, P.: A model building framework for answer set programming with external computations. TPLP **16**(4), 418–464 (2016)
2. Kaminski, T., Eiter, T., Inoue, K.: Exploiting answer set programming with external sources for meta-interpretive learning. CoRR **abs/1805.00068** (2018), http://arxiv.org/abs/1805.00068
3. Muggleton, S.H., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. Machine Learning **94**(1), 25–49 (2014)
4. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. Machine Learning **100**(1), 49–73 (2015)

[1] All instances can be found at http://www.kr.tuwien.ac.at/research/projects/inthex/hexmil/.