Towards a system based on asymmetric relative minimal generalisation

Stephen Muggleton, José Santos and Alireza Tamaddoni-Nezhad

Department of Computing,Imperial College London
Email: {shm,jcs06,atn}@doc.ic.ac.uk

Abstract. Over the last decade Inductive Logic Programming systems have been dominated by use of top-down refinement search techniques. In this paper we re-examine the use of bottom-up approaches to the construction of logic programs. In particular, we explore an asymmetric variant of Plotkin's Relative Least General Generalisation (RLGG) which is based on subsumption relative to a bottom clause. With Plotkin's RLGG, clause length grows exponentially in the number of examples. By contrast, in the Golem system, the length of ij-determinate RLGG clauses were shown to be polynomially bounded for given values of i and j. However, the determinacy restrictions made Golem inapplicable in many key application areas, including the learning of chemical properties from atom and bond descriptions. In this paper we show that with Asymmetric Relative Minimal Generalisations (or ARMGs) relative to a bottom clause, clause length is bounded by the length of the initial bottom clause. ARMGs , therefore do not need the determinacy restrictions used in Golem but still have a polynomial-time construction. An algorithm is described for constructing ARMGs together with some initial results on the clause lengths for some well-known ILP applications. In a longer follow-up paper we aim to extend the analysis and implementation of a system called ProGolem which combines bottom-clause construction in Progol with a Golem control strategy for constructing multiple clause predicate definitions.

1 Introduction

There are two key tasks at the heart of ILP systems: 1) enumeration of clauses which explain one or more of the positive examples and 2) evaluation of the numbers of positive and negative examples covered by these clauses. Top-down refinement techniques such as those found in [11, 8, 9], use a generate-and-test approach to problems 1) and 2). A new clause is first generated by application of a refinement step and then tested for coverage of positive and negative examples.

It has long been appreciated in AI [6] that generate-and-test procedures are less efficient than ones based on test-incorporation. The use of the bottom clause in Progol [4] represents a limited form of test-incorporation in which, by construction, all clauses in a refinement graph search are guaranteed to cover at least the example associated with the bottom clause. The use of Relative

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

Least General Generalisation (RLGG) in Golem [3] provides an extended form of test-incorporation in which constructed clauses are guaranteed to cover a given set of positive examples. However, in order to guarantee polynomial-time construction the form of RLGG in Golem was constrained to ij-determinate clauses. Without this constraint Plotkin [7] shows that the length of RLGG clauses grows exponentially in the number of positive examples covered.

In this paper we explore an asymmetric variant of Plotkin's RLGG which is based on subsumption order relative to a bottom clause [13]. We give a definition for Asymmetric Relative Minimal Generalisation (ARMGs) and show that the length of ARMGs is bounded by the length of the initial bottom clause. Hence, unlike in Golem, we do not need the determinacy restrictions to guarantee polynomial-time construction. However, we show that the resulting ARMG is not unique and that the operation is asymmetric. ARMG can easily be extended to the multiple example case by iteration. We show the outcome of initial experiments with an implementation of ARMG in which a bottom clause is progressively reduced to cover an increasing set of positive examples.

The paper is arranged as follows. In Section 2 we discuss subsumption relative to a bottom clause. ARMG is introduced in Section 3 and some of its properties are demonstrated. An algorithm for ARMG is given in Section 4 and an initial implementation of this algorithm is examined in Section 5. Section 6 concludes the paper.

2 Subsumption relative to bottom clause

In a previous paper [13] we introduced a subsumption order relative to a bottom clause and demonstrated how clause refinement in a Progol-like ILP system can be characterised with respect to this order. It was shown that, unlike for the general subsumption order, efficient least general generalisation operators can be designed for subsumption order relative to a bottom clause (i.e. $lgg_\perp$). In this section we briefly review the notion of subsumption order relative to bottom clause which is used in the definition of ARMG in this paper.

Definition 1 (Bottom clause $\perp_e$). *Let h, i be natural numbers, B be a set of Horn clauses, E be a set of positive and negative examples (ground unit clauses), M be a set of mode declarations and $\mathcal{L}_i(M)$ be a depth-bounded mode language as defined in [4]. Let e be a positive example in E. $\perp_e$ is the most specific clause in $\mathcal{L}_i(M)$ such that $B \wedge \perp_e \wedge \overline{e} \vdash_h \Box$ where $\vdash_h \Box$ denotes derivation of the empty clause in at most h resolutions.*

Definition 2 (Subsumption relative to $\perp$). *Let $\mathcal{L}_i(M)$ and $\perp_e$ be as defined in Definition 1 and C and D be clauses in $\mathcal{L}_i(M)$. We say C subsumes D relative to $\perp_e$, denoted by $C \succeq_\perp D$, iff there exist substitutions θ and σ such that θ induces a one-one correspondence for each literal l in C to a literal m in D such that $l\theta = m$ and likewise σ induces a one-one correspondence between all literals in D and a subset of literals in $\perp_e$.*

3 Asymmetric relative minimal generalisation

In this section we define a variant of Plotkin's RLGG which is based on subsumption order relative to a bottom clause. Plotkin's RLGG and the relative least general generalisation ($lgg_\perp$) in [13] were symmetric in the sense that $lgg_\perp(C, D) = lgg_\perp(D, C)$. However, the relative common generalisation considered in this paper is asymmetric and is denoted by $cg_\perp(C|D)$ and in general $cg_\perp(C|D) \neq cg_\perp(D|C)$. In the following we define asymmetric relative common generalisation and asymmetric relative minimal generalisation.

Definition 3 (Asymmetric relative common generalisation). *Let E, $\perp_e$ be as defined in Definition 1 and e and e' be positive examples in E. C is a common generalisation of e' relative to $\perp_e$, denoted by $C \in cg_\perp(e'|e)$, if $C \succeq_\perp \perp_e$ and C subsumes $\perp_{e'}$.*

Definition 4 (Asymmetric relative minimal generalisation). *Let E, $\perp_e$ be as defined in Definition 1, e and e' be positive examples in E and $cg_\perp(e'|e)$ be as defined in Definition 3. C is a minimal generalisation of e' relative to $\perp_e$, denoted by $C \in mg_\perp(e'|e)$, if $C \in cg_\perp(e'|e)$ and $C \succeq_\perp C' \in cg_\perp(e'|e)$ implies C is subsumption-equivalent to C' relative to $\perp_e$.*

The following theorem shows that the the length of ARMG is bounded by the length of $\perp_e$.

Theorem 1. *For each $C \in mg_\perp(e'|e)$ the length of C is bounded by the length of $\perp_e$.*

Proof. Let $C \in mg_\perp(e'|e)$. Then by definition $C \succeq_\perp \perp_e$ and according to Definition 2 there is a one-one correspondence between the literals of C and the literals of $\perp_e$. Hence, the length of C is bounded by the length of $\perp_e$. □

It follows from Theorem 1 that the number of literals in an ARMG is bounded by the length of $\perp_e$, which is shown in Theorem 26 in [4] to be polynomially bounded in the number of mode declarations for fixed values of i and j. Hence, unlike the RLGGs used in Golem, ARMGs do not need the determinacy restrictions and can be used in a wider range of problems including those which are non-determinate. The following theorem shows that ARMGs are not unique.

Theorem 2. *The set $mg_\perp(e'|e)$ can contain more than one clause which are not subsumption-equivalent relative to $\perp_e$.*

Proof. Let $M = \{p(+), q(+,-), r(+,-)\}$, $B = \{q(a,a), r(a,a), q(b,b), q(b,c), r(c,d)\}$, $e = p(a)$ and $e' = p(b)$. Then we have $\perp_e = p(X) \leftarrow q(X,X), r(X,X)$. The following clauses are both in $mg_\perp(e'|e)$: $C = p(X) \leftarrow q(X,X)$, $D = p(X) \leftarrow q(X,Y), r(Y,Z)$. C and D are minimal but not subsumption-equivalent relative to $\perp_e$. □

```
Asymmetric Relative Minimal Generalization (ARMG) Algorithm
    Input: Positive examples $e$, $e'$, mode declarations $M$, background knowledge $B$
      $C$ is $\bot_e = h \leftarrow b_1, .., b_n$
      While there is a blocking atom $b_i$ wrt $e'$ in the body of $C$
        Remove $b_i$ from $C$
        Remove atoms from $C$ which are not head-connected
      Repeat
    Output: $C$
```

Fig. 1. ARMG algorithm.

4 Algorithm for ARMG

An algorithm for constructing ARMGs is given in Figure 1. The following definitions are used to describe the ARMG algorithm.

Definition 5 (Head-connectness). *A definite clause $h \leftarrow b_1, .., b_n$ is said to be head-connected iff each body atom b_i contains at least one variable found either in h or in a body atom b_j, where $1 \leq j < i$.*

Definition 6 (Blocking atom). *Let B be background knowledge, E^+ the set of positive examples, $e \in E^+$ and $C = h \leftarrow b_1, \ldots, b_n$ be a definite clause. b_i is a blocking atom iff i is the least value such that for all θ, $e = h\theta, B \nvdash (b_1, \ldots, b_i)\theta$.*

The ARMG algorithm given in Figure 1 works by constructing the bottom clause associated with a particular positive example, and then dropping a minimal set of atoms from the body such that the bottom clause subsumes the bottom clause of a second positive example. Below we prove the correctness of the ARMG algorithm.

Theorem 3 (Correctness of ARMG). *Let B, M, E, $\bot_e$ be as defined in Definition 1, e and e' be positive examples in E, $mg_\bot(e'|e)$ be as defined in Definition 4 and $ARMG(e, e', M, B)$ as given in Figure 1. Then $C = ARMG(e, e', M, B)$ is in $mg_\bot(e'|e)$.*

Proof. Assume $C \notin mg_\bot(e'|e)$. In this case, either C is not a common generalisation of e and e' or it is not minimal. However, it is a common generalisation of e and e' since by construction it is a subset of $\bot_e$ in which all blocking literals with respect to e' are removed. So, C must be non-minimal. If C is non-minimal then it is subsumed by $C' \in mg_\bot(e'|e)$, which must either have literals not found in C or there is a substitution θ such that $C\theta = C'$. However, we have deleted the minimal set of literals, so it must be the second case. However, in the second case θ must be a renaming since the literals in C are all from $\bot_e$. Hence, C and C' are variants which contradicts the assumption and completes the proof. □

Figure 2 gives a comparison between Golem's determinate RLGG and the Asymmetric RMGs generated by the ARMG algorithm on Michalski's trains problem [5]. Note that Golem's RLGG cannot handle the predicate *has_car* because it is non-determinate. The first ARMG (2) subsumes the target concept which is eastbound(A) :- has_car(A,B), closed(B), short(B). Also, note that in this example RLGG (1) is shorter than ARMGs (2,3) since it only contains determinant literals.

1. $RLGG(e_1, e_2)) = RLGG(e_2, e_1)$ = eastbound(A) :- infront(A,B), short(B), open(B), shape(B,C), load(B,triangle,1), wheels(B,2), infront(B,D), shape(D, rectangle), load(D,E,1), wheels(D,F), infront(D,G), closed(G), short(G), shape(G,H), load(G,I,1), wheels(G,2).
2. $ARMG(e_1, e_2)$ = eastbound(A):- has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E), infront(A,E), closed(C), short(B), short(C), short(D), short(E), open(B), open(D), open(E), shape(B,F), shape(C,G), shape(D,F), shape(E,H), load(D,I,J),2), wheels(E,2)
3. $ARMG(e_2, e_1)$ = eastbound(A):- has_car(A,B), has_car(A,C), has_car(A,D), infront(A,D), closed(C), short(B), short(D), open(D), shape(B,E), shape(D,E), load(B,F,G), load(D,H,G), wheels(B,2), wheels(D,2)

Fig. 2. A comparison between Golem's determinate RLGG (1) and the Asymmetric ARMGs (2,3). Note that Golem's RLGG cannot handle the predicate *has_car* because it is non-determinate. The first ARMG (2) subsumes the target concept which is eastbound(A) :- has_car(A,B), closed(B), short(B).

5 Empirical evaluation

In this section we evaluate an implementation of the ARMG algorithm on different datasets. We compare the size and coverage of ARMGs as the number of examples used to construct the ARMGs is increased.

Materials The ARMG algorithm was implemented in YAP Prolog 5.1.4 and is publicly available at http://www.doc.ic.ac.uk/~jcs06. Several well-known ILP datasets have been used: Pyrimidines [1] is an examples of a determinate drug design problem. DSSTox [10], Carcinogenesis [12] and Alzheimers-Amine [2] are examples of non-determinate drug design problems, which therefore Golem cannot be applied to.

Methods For each of the datasets, a random sample of tuples of positive examples was selected. The tuple size, N, varied from 1 to 5 (i.e. samples of single examples, pairs of examples, triples, ...). The number of tuples in each sample is $N * |E^+|$ (i.e. sample size increases linearly with tuple size). For $N = 1$ the ARMG is simply the bottom clause. The following settings were used in the experiments: $i = 3$ for the bottom clause and $max.resolutions = 4 * |C|$ to test if an example is entailed by an ARMG. For each N in each dataset the average lengths of their ARMG as well as the ARMG positive coverage was computed. The lengths are reported as a fraction of the bottom clause length and the coverage is reported as a fraction of the total number of positive examples.

Results and discussion Results are summarised in Figure 3. In this figure short dashed lines represent the ARMG lengths and long dashed lines represent the ARMG coverages. When $N = 1$ the ARMG (i.e. bottom clause) coverage is almost invariably single example (a tiny fraction of the positive examples) and has maximum length. In general, the ARMG length follows an exponential decay and, symmetrically, the coverage has an exponential growth (since shorter clauses are more general). The rates of the exponential decay/growth vary with the dataset. In the longer version of this paper we will report full results (i.e. predictive accuracies and running times) for ProGolem, an ILP system which uses ARMGs as its hypotheses derivation mechanism.

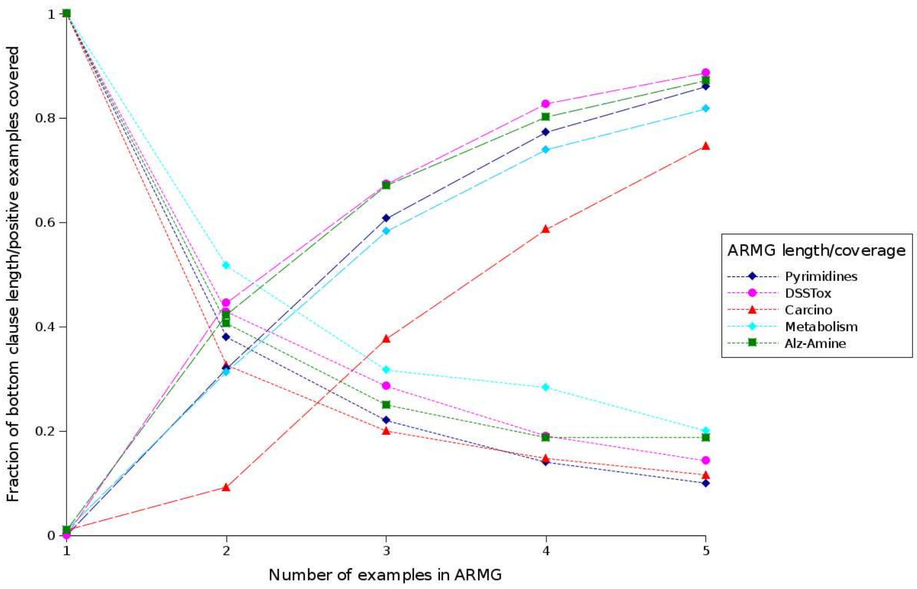

Fig. 3. ARMGs length (short dashed lines) and coverage (long dashed lines) as number of examples increases

6 Conclusions and further work

In this paper we have proposed an asymmetric variant of Plotkin's RLGG, called ARMG. In comparison to the determinate RLGGs used in Golem, ARMGs are capable of representing non-determinate clauses. Although this is also possible using Plotkin's RLGG, the cardinality of the Plotkin RLGG grows exponentially in the number of examples. By contrast, an ARMG is built by constructing a bottom clause for one example and then dropping a minimal set of literals to allow coverage of a second example. By construction the clause length is bounded by the length of the initially constructed bottom clause.

In the longer version of this paper we are aiming to provide details of a full implementation of a system called ProGolem. ProGolem will use the same general control structure as Golem, with ARMG in place of determinate RLGG. The use of top-down ILP algorithms such as Progol, tends to limit the maximum complexity of learned clauses, due to a search bias which favours simplicity. Long clauses generally require an overwhelming amount of search for systems like Progol and Aleph. We intend to explore whether ProGolem will have any advantages in situations when the clauses in the target theory are long and complex. Relevant applications which require such target theories include design and planning problems in which the target theory may involve complex conjunctions of literals. For instance, in the case of design, a single clause may be used to describe an entire digital circuit. Similarly, a plan may be described in a single clause. We believe that such targets should be more effectively learned by a bottom-up systems such as ProGolem since long clauses are easier to construct using bottom-up search.

References

1. R.D. King, S.H. Muggleton, R. Lewis, and M. Sternberg. Drug design by machine learning *Proc. of the Nat. Aca. of Sci.*, 89(23):11322–11326, 1992.
2. R.D. King, A. Srinivasan, and M.J.E. Sternberg. Relating chemical activity to structure *New Gen. Comp.*, 13:411–433, 1995.
3. S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Ind. Log. Prog.*, pages 281–298. Academic Press, 1992.
4. S.H. Muggleton. Inverse entailment and Progol. *New Gen. Comp.*, 13:245–286, 1995.
5. S. Muggleton. Progol datasets. http://www.doc.ic.ac.uk/∼shm/Software/progol4.2/, 1996.
6. N.J. Nilsson. *Principles of Artificial Intelligence.* Tioga, 1980.
7. G.D. Plotkin. *Automatic Methods of Inductive Inference.* PhD thesis, Edinburgh Univ., August 1971.
8. J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
9. L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proc. of the 13th Int. J. Conf. on AI.* MK, 1993.
10. A.M. Richard and C.R. Williams. Distributed structure-searchable toxicity (DSSTox) public database network: A proposal. *Mutation Research*, 499:27–52, 2000.
11. E.Y. Shapiro. *Algorithmic program debugging.* MIT Press, 1983.
12. A. Srinivasan, , R.D. King S.H. Muggleton, and M. Sternberg. Carcinogenesis predictions using ILP. In N.*Proc. of the 7th Int. Conf. on ILP*, pages 273–287. SV, 1997. LNAI 1297.
13. A. Tamaddoni-Nezhad and S.H. Muggleton. A note on refinement operators for IE-based ILP systems. In *Proc. of the 18th Int. Conf. on ILP*, LNAI 5194, pages 297–314. SV, 2008.