

COMSM0301: 2009/0 - Lecture 8 "Progol"

Oliver Ray
(oray@cs.bris.ac.uk)

Department of Computer Science
University of Bristol

1st February, 2010

Progol

- Bi-directional search:
 - constructs a bottom clause \perp from example e
 - searches top-down for a hypothesis h θ -subsuming \perp
- A*-like heuristic search to find maximally compressive clause
- Sequential covering

9 March, 2009

COMSM0301 - Progol

2

Bottom clause

- Wanted: clause h such that $B \wedge h \models e$
- Equivalent to: $B \wedge \neg e \models \neg h$
 - since e and h are single clauses, $\neg e$ and $\neg h$ are sets of skolemised ground literals, i.e. *countermodels*
- Let $\neg \perp$ be the set of all ground literals entailed by $B \wedge \neg e$
 - $\neg \perp$ is countermodel of any h such that $B \wedge h \models e$
 - may be infinite, and is restricted in Progol by mode declarations
- We have $B \wedge \neg e \models \neg \perp \models \neg h$, and thus $h \models \perp$
 - \perp is a de-skolemised clause
- Progol then searches for h that θ -subsumes \perp

9 March, 2009

COMSM0301 - Progol

3

Bottom clause: examples (1)

- $B = \{\text{hascar}(t1, c1), \text{short}(c1)\}$
 - $e = \text{eastbound}(t1)$
 - $\neg e = \{\neg \text{eastbound}(t1)\}$
 - $\neg \perp = \{\neg \text{eastbound}(t1), \text{hascar}(t1, c1), \text{short}(c1)\}$
 - $\perp = \text{eastbound}(t1) :- \text{hascar}(t1, c1), \text{short}(c1)$
-
- $B = \{\text{animal}(X) :- \text{pet}(X), \text{pet}(X) :- \text{dog}(X), \text{dog}(fido)\}$
 - $e = \text{nice}(fido)$
 - $\neg e = \{\neg \text{nice}(fido)\}$
 - $\neg \perp = \{\neg \text{nice}(fido), \text{dog}(fido), \text{pet}(fido), \text{animal}(fido)\}$
 - $\perp = \text{nice}(fido) :- \text{dog}(fido), \text{pet}(fido), \text{animal}(fido)$

9 March, 2009

COMSM0301 - Progol

4

Bottom clause: examples (2)

- $B = \{\text{animal}(X) :- \text{pet}(X), \text{pet}(X) :- \text{dog}(X)\}$
 - $e = \text{nice}(fido) :- \text{dog}(fido)$
 - $\neg e = \{\neg \text{nice}(fido), \text{dog}(fido)\}$
 - $\neg \perp = \{\neg \text{nice}(fido), \text{dog}(fido), \text{pet}(fido), \text{animal}(fido)\}$
 - $\perp = \text{nice}(fido) :- \text{dog}(fido), \text{pet}(fido), \text{animal}(fido)$
-
- $B = \{\text{animal}(X) :- \text{pet}(X), \text{pet}(X) :- \text{dog}(X)\}$
 - $e = \text{nice}(X) :- \text{dog}(X)$
 - $\neg e = \{\neg \text{nice}(c_x), \text{dog}(c_x)\}$
 - $\neg \perp = \{\neg \text{nice}(c_x), \text{dog}(c_x), \text{pet}(c_x), \text{animal}(c_x)\}$
 - $\perp = \text{nice}(X) :- \text{dog}(X), \text{pet}(X), \text{animal}(X)$

9 March, 2009

COMSM0301 - Progol

5

Mode declarations

- Head modes
 - `modeh(*, rev(+list, -list))`
- Body modes
 - `modeb(*, rev(+list, -list))`
 - `modeb(*, app(+list, [+int], -list))`
 - `modeb(*, +list=[-int]-list)`
 - `modeb(*, +any=#any)`
- Background knowledge
 - `app([], Ys, Ys).`
 - `app([X|Xs], Ys, [X|Zs]) :- app(Xs, Ys, Zs).`
 - `rev([], []).`
 - + type definitions

9 March, 2009

COMSM0301 - Progol

7

Bottom clause algorithm (head)

```

e = a:-b1,...,bn;
¬e = ¬a ∧ b1 ∧ ... ∧ bn added to B;
InTerms = ∅; ⊥ = ∅;
h = first head mode declaration such that hθ = a;
FOR EACH v→t ∈ θ
  IF type(v) = # THEN replace v in h by t
  ELSE replace v in h by vt;
  IF type(v) = + THEN InTerms = InTerms ∪ {t};
  ⊥ = ⊥ + {h};
    
```

9 March, 2009

COM60301 - Prolog

8

Bottom clause algorithm (body)

```

REPEAT until variable depth has been reached
FOR EACH body mode declaration b
  FOR EACH subs. θ of +vars with terms from InTerms
    REPEAT recall(b) TIMES
      call goal bθ with answer θ';
      FOR EACH v→t ∈ θ ∪ θ'
        IF type(v) = # THEN replace v in b by t
        ELSE replace v in b by vt;
        IF type(v) = - THEN InTerms = InTerms ∪ {t};
        ⊥ = ⊥ + {-b};
      Increase variable depth
    
```

9 March, 2009

COM60301 - Prolog

9

Heuristic search

- Parameters to evaluate a clause c:
 - p(c) = number of positive examples covered
 - n(c) = number of negative examples covered
 - g(c) = length of clause
 - h(c) = number of atoms needed to define all output variables
- Evaluation function: $f = (p-n) - (g+h)$
 - higher f is better
- A*-like search
 - stop refining when n=0
 - terminate when all clauses on the agenda have lower f-value than current best

9 March, 2009

COM60301 - Prolog

10

Prolog settings

- **set(c, 4)?** max. length of clause
- **set(h, 30)?** resolution bound for constructing bottom clause
- **set(i, 3)?** max. variable depth
- **set(inflate, 100)?** weight of positive (percentage)
- **set(nodes, 10000)?** max.nr. of candidate hypotheses searched
- **set(noise, 0)?** allowable percentage of negatives covered
- **set(memoing, on)?** memorise and re-use statistics
- **set(searching, off)?** changes behaviour of **example!**
- **set(verbose, 2)?** verbosity of output (between 0 and 2)
- **set(posonly, off)?** learning from positive examples only

9 March, 2009

COM60301 - Prolog

11

Additional Prolog features

- Integrity constraints
 - used to discard single hypotheses
 - e.g. hypotheses should only cover person:
 - false: ¬hypothesis(male(X), Body, _) , Body, not person(X).
- Prune statements
 - express further declarative bias
 - e.g. exclude mutual recursion:
 - `prune(Head, Body) :- in(Atom, Body), clause(Atom, Body1), in(Head, Body1).`

9 March, 2009

COM60301 - Prolog

12

Learning from positive examples

- Bayesian/MDL perspective: trade off size and generality of a hypothesis
- In order to determine generality:
 - construct Stochastic Logic Program from modeh declaration
 - `modeh(1, move(+piece, pos(+file, +rank), pos(+file, +rank)))`
 - `move(A, pos(B, C), pos(D, E)) :- piece(A), file(B), rank(C), file(D), rank(E)`
 - estimate probabilities from examples
 - generate random sample of instances from SLP and estimate the probability that random instance is covered

9 March, 2009

COM60301 - Prolog

13

Example

%Mode Declarations

```
:- modeh(*,m(-int,+list))?  
:- modeb(*,m(-int,+list))?  
:- modeb(*,+list=[-int]-list)?
```

% Background Knowledge

```
m(1,[2,1]).
```

% Examples

```
m(1,[1,2,1]).  
:-m(1,[]).
```

Bottom clause = $m(X,Y) :- Y=[Q|R], m(Q,R) R=[S|T], T=[Q|U]$

InTerms = $\{[1,2,1],1,[2,1],2,[1,[]]\}$

Hypothesis = $m(X,Y) :- Y=[Q|R]$