Logic Programming in a Nutshell

Wamberto Vasconcelos – SSP Group

15 June 2000

Organisation

1. Syntax of Logic Programs
2. Unification
3. Resolution
4. Procedural $\times$ Declarative Meaning of LPs
5. Proof (search) trees
6. Logic Programming $\times$ Prolog
7. More details...

1. Syntax of Logic Programs

- *Terms* (t_i):
  - *constants*: a, b, c;
  - *variables*: x, y, z;
  - *functions*: $f(t_1, \ldots, t_n)$;

- *Literals* (l_i):
  - *positive*: $p(t_0, \ldots, t_n)$ (*atom*)
  - *negative*: $\neg p(t_0, \ldots, t_n)$

- *Clauses*:

$$l_1 \vee \cdots \vee l_n$$

If $A_i, B_j, 1 \leq i \leq n, 1 \leq j \leq m$, are atoms:

$$A_1 \vee \cdots \vee A_n \vee \neg B_1 \vee \cdots \vee \neg B_m$$

$$A_1, \ldots, A_n \leftarrow B_1, \ldots, B_m$$

  - *Program Clause*: $\boxed{\underbrace{A}_{\text{head}} \leftarrow \underbrace{B_1, \ldots, B_m}_{\text{body}}}$
  - *Unit Clause*: $\boxed{A \leftarrow}$

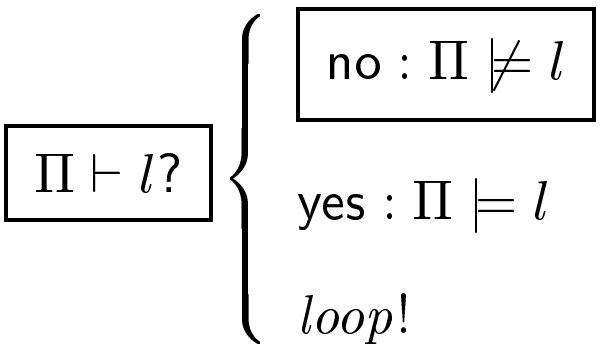
- *Logic Program*: set of program and unit clauses

2. Unification

- Means to compute values in LP
- Variables x_i can be associated with terms t_i via *substitutions*: $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$
- $t = f(x, a, g(z)), \theta = \{x/1, z/2\}, t\theta = f(1, a, g(2))$.
  (*application* of θ to t)
- Def.: Given terms s and t, find θ such that $s\theta = t\theta$
  (θ *unifies* s and t)
- Two atoms unify if they are of the form
  $p(s_1, \ldots, s_n)$ $\quad$ $p(t_1, \ldots, t_n)$
  and s_i and t_i, $1 \leq i \leq n$, unify.
- *Most Generic Unifier* (*MGU*): simplest θ
  (without unnecessary x_i/t_i)
- Different algorithms to find MGU's
- Example: $p(a, f(x, y), b)\theta = p(z, f(g(z), g(z)), z)\theta$?

3. Resolution

- Problem: given Π and l, $\Pi \vdash l$?
- Automated Theorem Proving
- Resolution is *refutation* procedure (semi-decidable)



- Algorithm:
  input program Π and query $\leftarrow G_1, \ldots, G_n$
  output $(G_1, \ldots, G_n)\theta$ such that $\Pi \vdash (G_1, \ldots, G_n)\theta$

```
SLDRes(Π, (G_1, ..., G_n))
begin
    choose G_i
    choose (A ← B_1, ..., B_m) ∈ Π such that MGU(G_i, A, θ)
    SLDRes(Π, (G_1, ..., G_{i-1}, B_1, ..., B_m, G_{i+1}, ..., G_n)θ)
end
```

- SLD-Resolution is *sound* and *complete*

4. Procedural $\times$ Declarative Meaning of LPs

- Procedural: actual computation (derivation)
- Declarative: potential computations (theory)
- Example: $\Pi = \boxed{\begin{array}{l} p(x) \leftarrow p(y) \\ p(a) \leftarrow \end{array}}$

  Derivation of $\leftarrow p(a)$:

  1. choose $p(x') \leftarrow p(y')$ (N.B. variation!)
     $MGU(p(a), p(x'), \{x'/a\})$
  2. find derivation for $\leftarrow p(y')$
  2.1 choose $p(x'') \leftarrow p(y'')$ (N.B. another variation!)
     $MGU(p(x'), p(x''), \{x'/x''\})$
  2.2 find derivation for $\leftarrow p(y'')$
  2.2.1 choose $p(a) \leftarrow$
     $MGU(p(y''), p(a), \{y''/a\})$

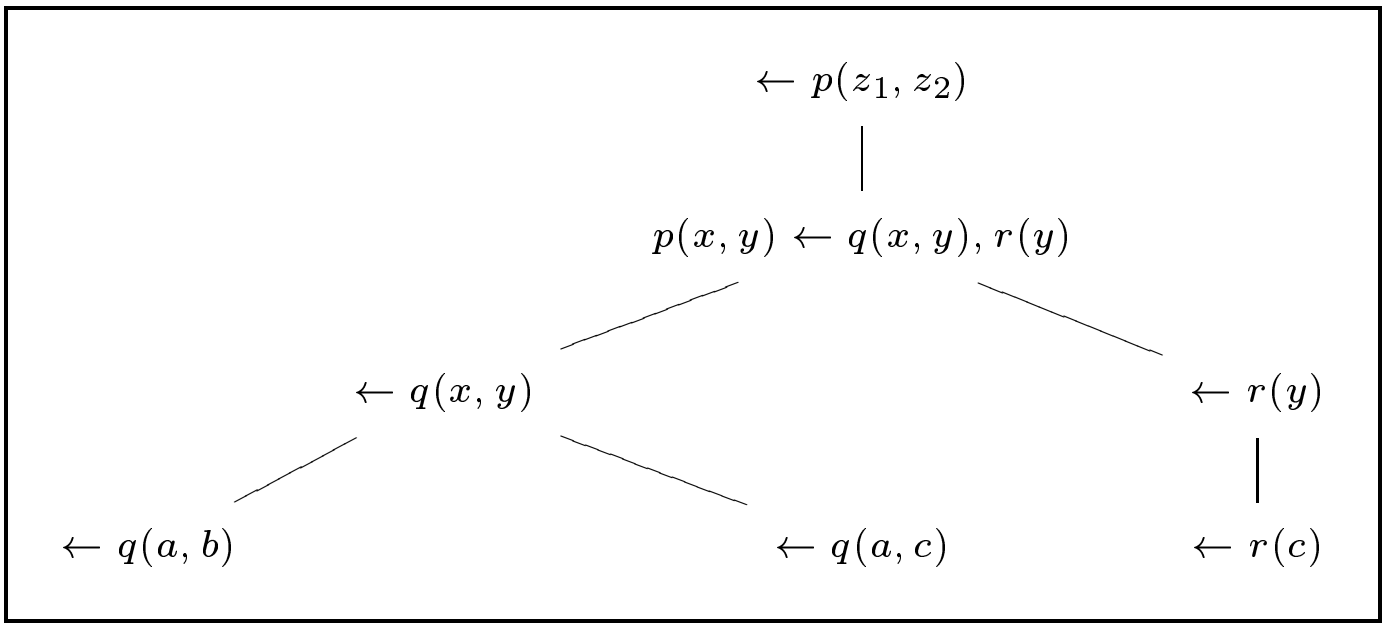
  Declarative meaning:

$$(p(a) \leftarrow) \in Th(\Pi)$$

5. Proof Trees

- Solution space as a tree
- Each choice point (**choose**) yields branches in tree
- $\Pi = \boxed{\begin{array}{l} p(x,y) \leftarrow q(x,y), r(y) \\ q(a,b) \leftarrow \\ q(a,c) \leftarrow \\ r(c) \leftarrow \end{array}}, \leftarrow p(z_1, z_2)$

  Proof Tree:



6. Logic Programming $\times$ Prolog

- Choice points in LP are *non-deterministic*
- Choice points in Prolog are *deterministic*:

  literals – left to right;
  clauses – top to bottom

- Programmers should be aware of Prolog's shortcomings:

$$\boxed{\begin{array}{l} p(x) \leftarrow p(y) \\ p(a) \leftarrow \end{array}} \not\vdash_{\text{Prolog}} p(a)$$

- Proof tree: depth-first – advantages and drawbacks
- Metaprogramming to overcome limitations:
  - breadth-first search in proof-tree
  - iterative deepening

7. More Details...

- Links:
  - `http://www.logic-programming.org`
  - `http://www.cwi.nl/projects/alp/`

- Papers:
  - "Predicate Logic as a Programming Language"
    R. A. Kowalski, *Inf. Proc.*, 1974, North-Holland
  - "Logic Programming"
    K. R. Apt, *Handbook of Theor. Comp. Sci.*, 1990, Elsevier
  - "Logic and Logic Programming"
    J. A. Robinson, *Comm. ACM*, 35(3):40–65, 1992

- Books:
  - *Foundations of Logic Programming*
    J. W. Lloyd (1984, Springer-Verlag)
  - *From Logic to Logic Programming*
    K. Doets (1994, MIT Press)
  - *From Logic Programming to Prolog*
    K. R. Apt (1996, Prentice-Hall)