Definite Logic Programs: Derivation and Proof Trees

CSE 505 – Computing with Logic

Stony Brook University

http://www.cs.stonybrook.edu/~cse505

Refutation in Predicate Logic

parent(pam, bob).
parent(tom, bob).
parent(tom, liz). …

anc(X,Y) :- parent(X,Y).
anc(X,Y) :- parent(X,Z),
          anc(Z,Y).

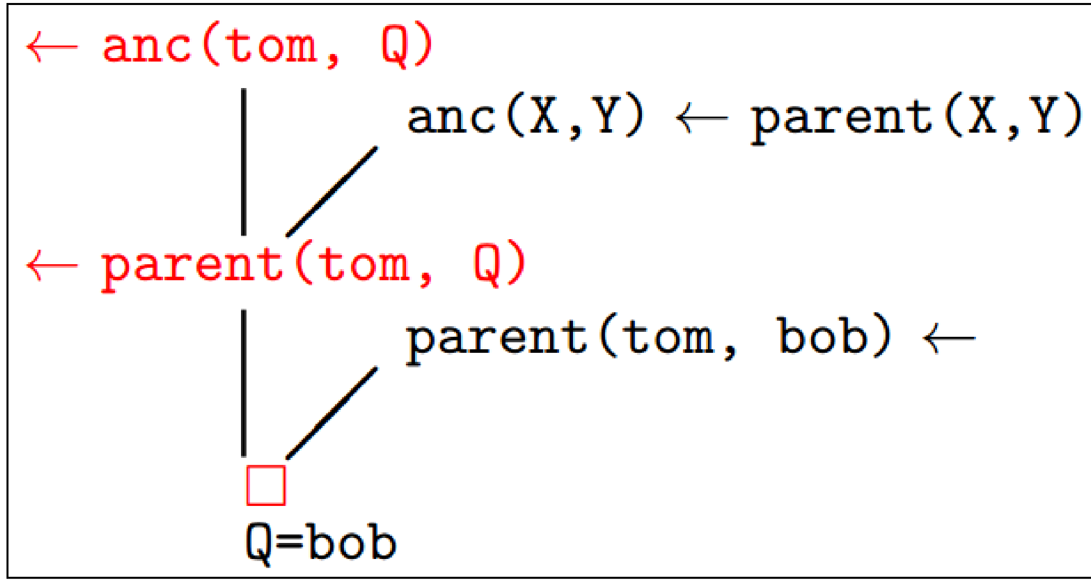
- For what values of Q is anc(tom,Q) a logical consequence of the above program?
- Negate the goal F: i.e. $\neg F = \forall Q.\neg anc(tom, Q)$.
- Consider the clauses in $P \cup \neg F$
  - Note that a program clause written as p(A,B) :- q(A,C), r(B,C) can be rewritten as: $\forall A, B, C\ (p(A, B) \lor \neg q(A, C) \lor \neg r(B, C))$

  I.e., l.h.s. literal is positive, while all r.h.s. literals are negative
  - Note also that all variables are universally quantified in a clause!

(c) Paul Fodor (CS Stony Brook) and Elsevier

Refutation: An Example

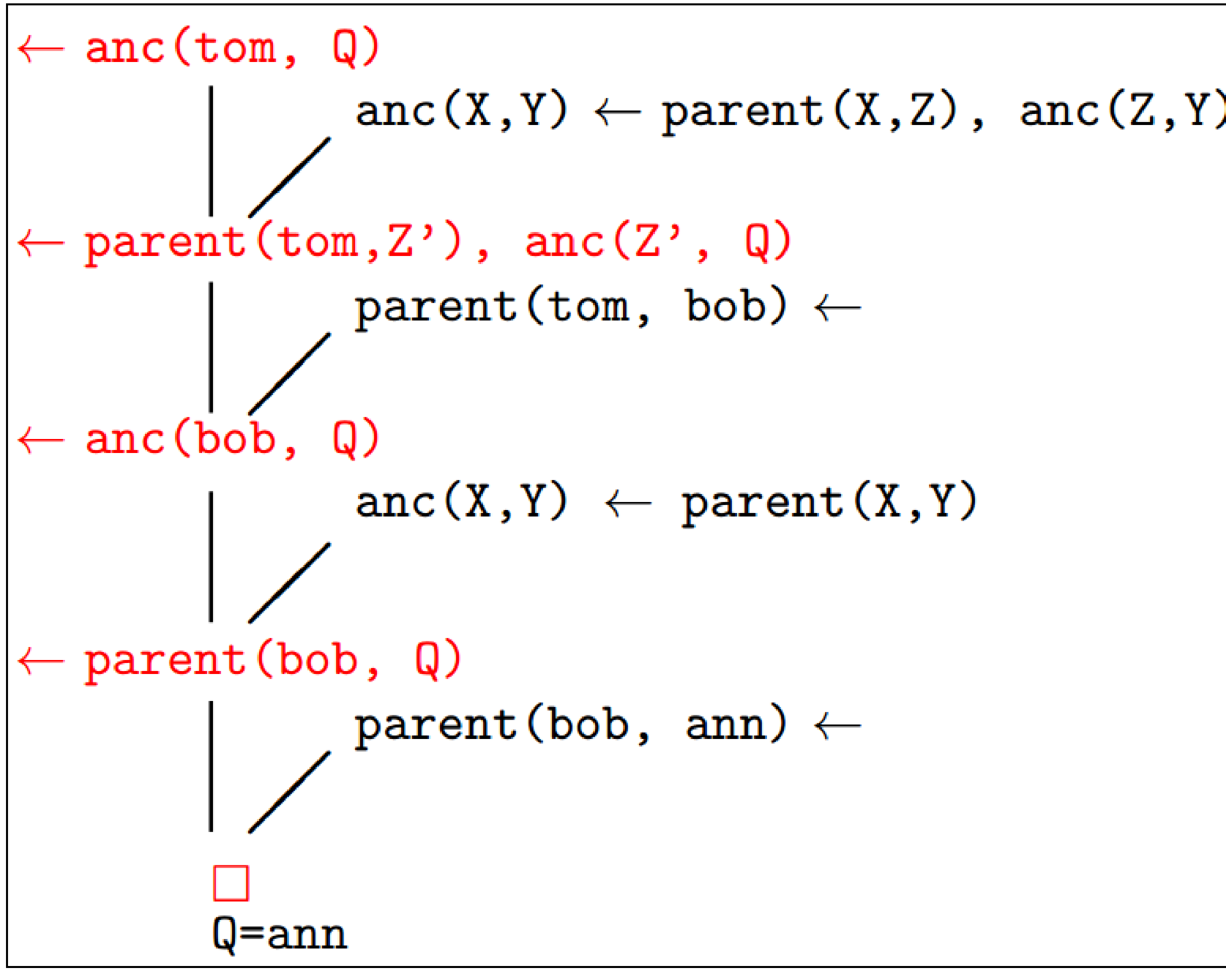
```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
anc(X,Y) :-
        parent(X,Y).
anc(X,Y) :-
        parent(X,Z),
        anc(Z,Y).
```



(c) Paul Fodor (CS Stony Brook) and Elsevier

Refutation: An Example

```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).
anc(X,Y) :-
        parent(X,Y).
anc(X,Y) :-
        parent(X,Z),
        anc(Z,Y).
```



(c) Paul Fodor (CS Stony Brook) and Elsevier

Unification

- Operation done to "match" the goal atom with the head of a clause in the program.
- Forms the basis for the ***matching*** operation we used for Prolog evaluation.
  - f(a,Y) and f(X,b) unify when X=a and Y=b
  - f(a,X) and f(X,b) do not unify

(c) Paul Fodor (CS Stony Brook) and Elsevier

Substitutions

- A substitution is a mapping between variables and values (terms)
  - Denoted by {X1/t1, X2/t2, …, Xn/tn} such that
    - Xi ≠ ti , and
    - Xi and Xj are distinct variables when i ≠ j.
  - The empty substitution is denoted by ε (or {}).
  - A substitution is said to be a ***renaming*** if it is of the form {X1/Y1, …, Xn/Yn} and Y1, …,Yn is a permutation of X1, …, Xn.
    - Example: {X/Y ,Y /X} is a renaming substitution.

(c) Paul Fodor (CS Stony Brook) and Elsevier

Substitutions and Terms

- Application of a substitution:
  - $X\theta = t$ if $X/t \in \theta$.
  - $X\theta = X$ if $X/t \notin \theta$ for any term t.
  - Application of a substitution $\{X1/t1, \ldots, Xn/tn\}$ to a ***term/formula*** F:
    - is a term/formula obtained by simultaneously replacing every ***free*** occurrence of Xi in F by ti .
    - Denoted by $F\theta$ [and $F\theta$ is said to be an instance of F]
- Example:

$$p(f(X, Z), f(Y, a))\{X/g(Y), Y/Z, Z/a\} = p(f(g(Y), a), f(Z, a))$$

(c) Paul Fodor (CS Stony Brook) and Elsevier

Composition of Substitutions

- Composition of substitutions $\theta = \{X1/s1, \ldots, Xm/sm\}$ and $\sigma = \{Y1/t1, \ldots, Yn/tn\}$:
  - First form the set $\{X1/s1\sigma, \ldots, Xm/sm\sigma, Y1/t1, \ldots, Yn/tn\}$
  - Remove from the set $Xi/si\sigma$ if $si\sigma = Xi$
  - Remove from the set Yj/tj if Yj is identical to some variable Xi
- Example: Let $\theta = \sigma = \{X/g(Y), Y/Z, Z/a\}$. Then $\theta\sigma = \{X/g(Y), Y/Z, Z/a\}\{X/g(Y), Y/Z, Z/a\} = \{X/g(Z), Y/a, Z/a\}$
- More examples: Let $\theta = \{X/f(Y)\}$ and $\sigma = \{Y/a\}$
  - $\theta\sigma = \{X/f(a), Y/a\}$
  - $\sigma\theta = \{Y/a, X/f(Y)\}$
- Composition is not commutative but is associative: $\theta(\sigma\gamma) = (\theta\sigma)\gamma$
- Also, $E(\theta\sigma) = (E\theta)\sigma$

(c) Paul Fodor (CS Stony Brook) and Elsevier

Idempotence

- A substitution θ is ***idempotent*** iff $\theta\theta = \theta$.
- Examples:
  - $\{X/g(Y), Y/Z, Z/a\}$ is not idempotent since

    $\{X/g(Y), Y/Z, Z/a\}\{X/g(Y), Y/Z, Z/a\} = \{X/g(Z), Y/a, Z/a\}$
  - $\{X/g(Z), Y/a, Z/a\}$ is not idempotent either since

    $\{X/g(Z), Y/a, Z/a\}\{X/g(Z), Y/a, Z/a\} = \{X/g(a), Y/a, Z/a\}$
  - $\{X/g(a), Y/a, Z/a\}$ is idempotent
- For a substitution $\theta = \{X1/t1, \ldots, Xn/tn\}$,
  - $\mathrm{Dom}(\theta) = \{X1, X2, \ldots Xn\}$
  - $\mathrm{Range}(\theta)$ = set of all variables in t1, …, tn
- A substitution θ is ***idempotent*** iff $\mathrm{Dom}(\theta) \cap \mathrm{Range}(\theta) = \emptyset$

(c) Paul Fodor (CS Stony Brook) and Elsevier

Unifiers

- A substitution θ is a ***unifier*** of two terms s and t if $s\theta$ is identical to $t\theta$
- θ is a unifier of a set of equations $\{s1 = t1, \ldots, sn = tn\}$, if for all i,

  $si\ \theta = ti\ \theta$

- A substitution θ is more general than σ (written as $\theta \geq \sigma$) if there is a substitution ω such that $\sigma = \theta\omega$
- A substitution θ is a ***most general unifier*** (mgu) of two terms (or a set of equations) if for every unifier σ of the two terms (or equations) $\theta \geq \sigma$
- Example: Consider two terms f(g(X), Y, a) and f(Z, W , X).

  $\theta 1 = \{X/a, Y/b, Z/g(a), W/b\}$ is a unifier

  $\theta 2 = \{X/a, Y/W , Z/g(a)\}$ is also a unifier

  $\theta 2$ is a most general unifier

(c) Paul Fodor (CS Stony Brook) and Elsevier

Equations and Unifiers

- A set of equations E is in ***solved form*** if it is of the form $\{X1 = t1, \ldots, Xn = tn\}$ iff no Xi appears in any tj.
  - Given a set of equations $E = \{X1 = t1, \ldots, Xn = tn\}$ the substitution $\{X1/t1, \ldots, Xn/tn\}$ is an idempotent mgu of E
- Two sets of equations E1 and E2 are said to be ***equivalent*** iff they have the same set of unifiers.
- To find the mgu of two terms s and t, try to find a set of equations in solved form that is equivalent to $\{s = t\}$.

If there is no equivalent solved form, there is no mgu.

(c) Paul Fodor (CS Stony Brook) and Elsevier

A Simple Unification Algorithm (via Examples)

- Example 1: Find the mgu of $f(X, g(Y))$ and $f(g(Z), Z)$

$$\{f(X, g(Y)) = f(g(Z), Z)\} \Rightarrow \{X = g(Z), g(Y) = Z\}$$
$$\Rightarrow \{X = g(Z), Z = g(Y)\}$$
$$\Rightarrow \{X = g(g(Y)), Z = g(Y)\}$$

- Example 2: Find the mgu of $f(X, g(X), b)$ and $f(a, g(Z), Z)$

$$\{f(X, g(X), b) = f(a, g(Z), Z)\} \Rightarrow \{X = a, g(X) = g(Z), b = Z\}$$
$$\Rightarrow \{X = a, g(a) = g(Z), b = Z\}$$
$$\Rightarrow \{X = a, a = Z, b = Z\}$$
$$\Rightarrow \{X = a, Z = a, b = Z\}$$
$$\Rightarrow \{X = a, Z = a, b = a\}$$
$$\Rightarrow \text{fail}$$

(c) Paul Fodor (CS Stony Brook) and Elsevier

A Simple Unification Algorithm

Given a set of equations E:

repeat

select s = t ∈ E;

case s = t of

1. f(s1, …,sn) = f(t1, …,tn):
   replace the equation by si = ti for all i
2. f(s1, …,sn) = g(t1, …,tm), f ≠ g or n ≠ m:
   halt with failure
3. X = X : remove the equation
4. t = X : where t is not a variable
   replace equation by X = t
5. X = t : where X ≠ t and X occurs more than once in E:
   if X is a proper subterm of t
   then halt with failure (5a)
   else replace all other X in E by t (5b)

until no action is possible for any equation in E

return E

(c) Paul Fodor (CS Stony Brook) and Elsevier

A Simple Unification Algorithm

Example: Find the mgu of $f(X, g(Y))$ and $f(g(Z), Z)$

$\{f(X, g(Y)) = f(g(Z), Z)\}$

$\Rightarrow \{X = g(Z), g(Y) = Z\}$ case 1

$\Rightarrow \{X = g(Z), Z = g(Y)\}$ case 4

$\Rightarrow \{X = g(g(Y)), Z = g(Y)\}$ case 5b

(c) Paul Fodor (CS Stony Brook) and Elsevier

A Simple Unification Algorithm

Example: Find the mgu of f(X, g(X)) and f(Z, Z)

$$\{f(X, g(X)) = f(Z, Z)\}$$

$\Rightarrow \{X = Z, g(X) = Z\}$	case 1
$\Rightarrow \{X = Z, g(Z) = Z\}$	case 5b
$\Rightarrow \{X = Z, Z = g(Z)\}$	case 4
$\Rightarrow$ fail	case 5a

(c) Paul Fodor (CS Stony Brook) and Elsevier

Complexity of the unification algorithm

- Consider $E=\{g(X1, \ldots, Xn)=g(f(X0, X0), f(X1, X1), \ldots, f(Xn-1, Xn-1)\}$
  - By applying case 1 of the algorithm, we get $\{X1 = f(X0, X0), X2 = f(X1, X1), X3 = f(X2, X2), \ldots, Xn = f(Xn-1, Xn-1)\}$
  - If terms are kept as trees, the final value for Xn is a tree of size $O(2^n)$.
  - Recall that for case 5 we need to first check if a variable appears in a term, and this could now take $O(2^n)$ time.
  - There are linear-time unification algorithms that share structures (terms as DAGs).
  - X = t is the most common case for unification in Prolog. The fastest algorithms are linear in t.
  - Prolog cuts corners by omitting case 5a (the occur check), thereby doing X = t in constant time.

(c) Paul Fodor (CS Stony Brook) and Elsevier

Most General Unifiers

- Note that mgu stands for **a** most general unifier.
- There may be more than one mgu. E.g. f (X) = f (Y) has two mgus:
  - $\{X / Y\}$
  - $\{Y / X\}$
- If θ is an mgu of s and t, and ω is a renaming, then $\theta\omega$ is an mgu of s and t.
- If θ and σ are mgus of s and t, then there is a renaming ω such that $\theta = \sigma\omega$.
  - MGU is unique up to renaming

(c) Paul Fodor (CS Stony Brook) and Elsevier

SLD Resolution

- Selective Linear Definite clause Resolution:

$$\frac{\leftarrow A_1, \ldots, A_{i-1}, \boxed{A_i}, A_{i+1}, \ldots, A_m \quad B_0 \leftarrow B_1, \ldots, B_n}{\leftarrow (A_1, \ldots, A_{i-1}, \boxed{B_1, \ldots, B_n}, A_{i+1}, \ldots, A_m)\theta}$$

where:

1. Aj are atomic formulas
2. B0 ← B1, ..., Bn is a (renamed) definite clause in the program
3. θ = mgu(Ai , B0)
   - Ai is called the selected atom
   - Given a goal ← A1, ..., An a function called the selection function or computation rule selects Ai

(c) Paul Fodor (CS Stony Brook) and Elsevier

SLD Resolution (cont.)

- When the resolution rule is applied, from a goal G and a clause C, we get a new goal G'
- We then say that G' is derived directly from G and C:

$$G \overset{C}{\rightsquigarrow} G'$$

- An ***SLD Derivation*** is a sequence

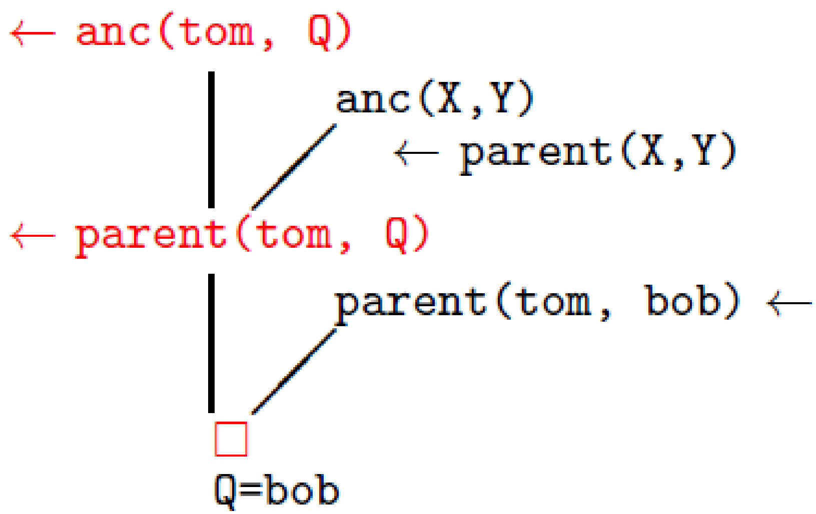
$$G_0 \overset{C_0}{\rightsquigarrow} G_1 \cdots G_i \overset{C_i}{\rightsquigarrow} G_{i+1} \cdots$$

(c) Paul Fodor (CS Stony Brook) and Elsevier

Refutation & SLD Derivation

```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

anc(X,Y) :-
   parent(X,Y).
anc(X,Y) :-
   parent(X,Z),
   anc(Z,Y).
```



$$\texttt{anc(tom, Q)} \leadsto \texttt{parent(tom, Q)} \leadsto \square$$

(c) Paul Fodor (CS Stony Brook) and Elsevier

Refutation & SLD Derivation

```
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(pat, jim).

anc(X,Y) :-
   parent(X,Y).
anc(X,Y) :-
   parent(X,Z),
   anc(Z,Y).
```

```
← anc(tom, Q)
                    anc(X,Y)
                      ← parent(X,Z), anc(Z,Y)
← parent(tom,Z'), anc(Z', Q)
                    parent(tom, bob) ←
← anc(bob, Q)
                    anc(X,Y)
                      ← parent(X,Y)
← parent(bob, Q)
                    parent(bob, ann) ←
□
Q=ann
```

```
anc(tom, Q)
⇝ parent(tom, Z')
        anc(Z', Q)
⇝ anc(bob, Q)
⇝ parent(bob, Q)
⇝ □
```


(c) Paul Fodor (CS Stony Brook) and Elsevier

Computed Answer Substitution

- Let θ0, θ1, . . . , θn−1 be the sequence of mgus used in derivation

$$G_0 \overset{C_0}{\leadsto} G_1 \cdots G_{n-1} \overset{C_{n-1}}{\leadsto} G_n$$

Then θ=θ0θ1 · · · θn−1 is the computed subtitution of the derivation

- Example:

Goal	Clause Used	mgu
`anc(tom, Q)`	`anc(X',Y') :-`	
	`parent(X',Z'), anc(Z',Y')`	$\theta_0 = \{X'/\mathtt{tom}, Y'/Q\}$
`parent(tom, Z'),`		
`anc(Z', Q)`	`parent(tom, bob).`	$\theta_1 = \{Z'/\mathtt{bob}\}$
`anc(bob, Q)`	`anc(X'', Y'') :-`	
	`parent(X'', Y'').`	$\theta_2 = \{X''/\mathtt{bob}, Y''/Q\}$
`parent(bob, Q)`	`parent(bob, ann).`	$\theta_3 = \{Q/\mathtt{ann}\}$
□		

- Computed substitution for the above derivation is

θ0θ1θ2θ3 = {X'/tom, Y'/ann, Z'/bob, X''/bob, Y''/ann, Q/ann}

(c) Paul Fodor (CS Stony Brook) and Elsevier

Computed Answer Substitution

- A finite derivation of the form

$$G_0 \overset{C_0}{\leadsto} G_1 \cdots G_{n-1} \overset{C_{n-1}}{\leadsto} G_n$$

where Gn = (i.e., an empty goal) is an ***SLD refutation*** of G0

- The computed substitution of an SLD refutation of G, restricted to variables of G, is a ***computed answer substitution*** for G.
- Example (contd.): The computed answer substitution for the above SLD refutation is

{X'/tom, Y'/ann, Z'/bob, X''/bob, Y''/ann, Q/ann}

restricted to Q:

{Q/ann}

(c) Paul Fodor (CS Stony Brook) and Elsevier

Failed SLD Derivation

- *A derivation of a goal clause G0 whose last element is not empty, and cannot be resolved with any clause of the program.*
- Example: consider the following program:

```
grandfather(X,Z) :- father(X,Y), parent(Y,Z).
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
father(a,b).
mother(b,c).
```

- A derivation of grandfather(a,Q) is:

```
⇝  father(a,Y'), parent(Y',Q)
⇝  parent(b,Q)
⇝  father(b,Q)
```

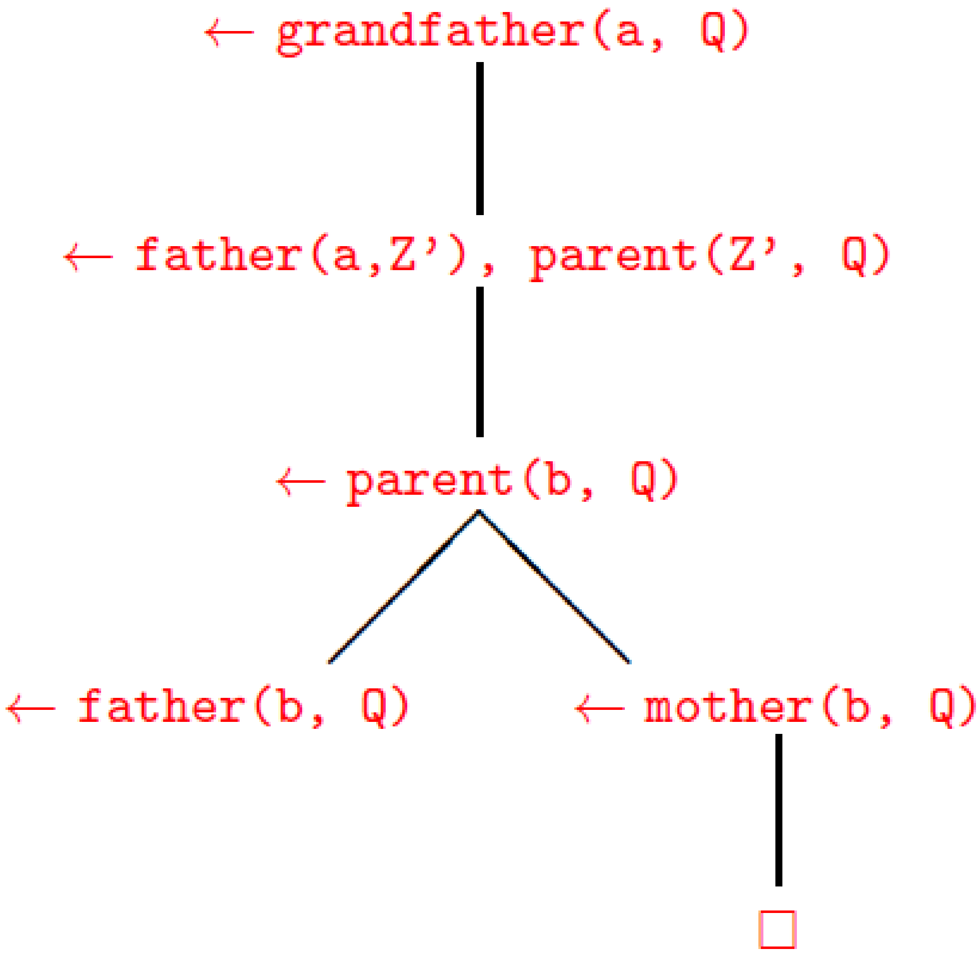
SLD Tree

- A tree where every path is an SLD derivation

```
grandfather(X,Z) :-
  father(X,Y), parent(Y,Z).

parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).

father(a,b).
mother(b,c).
```



(c) Paul Fodor (CS Stony Brook) and Elsevier

Soundness of SLD resolution

- Let P be a definite program, R be a computation rule, and θ be a computed answer substitution for a goal G. Then $\forall G\theta$ is a logical consequence of P.
- Proof is by induction on the number of resolution steps used in the refutation of G.
- Base case uses the following lemma:
  - Let F be a formula and F' be an instance of F, i.e. $F' = F\theta$ for some substitution θ.
    Then $(\forall F) \models (\forall F')$.

(c) Paul Fodor (CS Stony Brook) and Elsevier