Logic Programming in Prolog
Part II
Substitutions, SLD Resolution
Prolog Execution

Raj Sunderraman

Substitutions

Definition: A <u>substitution</u> is a finite set of pairs of terms, $\{X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n\}$, where each t_i is a term and each X_i is a variable such that X_i != t_i and X_i != X_j if i != j. Empty substitution is represented by ε

$\theta = \{X \leftarrow 20, Y \leftarrow f(a), Z \leftarrow [1,2,3]\}$

Definition: The <u>application</u> $X\theta$ of a substitution θ to a variable X is defined as follows:

$X\theta = t$ if X := t is in θ
$= X$ otherwise

$Y \{X \leftarrow 20, Y \leftarrow f(a), Z \leftarrow [1,2,3]\} = f(a)$
$U \{X \leftarrow 20, Y \leftarrow f(a), Z \leftarrow [1,2,3]\} = U$

Definition: Let θ be a substitution $\{X_1 \leftarrow t_1, \ldots, X_n \leftarrow t_n\}$ and E is a term or formula. Then, <u>application</u> of θ to E, $E\theta$ is a term or formula obtained from E by simultaneously replacing every occurrence of X_i by t_i , 1<= i<=n. $E\theta$ is called an *instance* of E.

$p(f(X,Z),f(Y,a)) \{X \leftarrow a, Y \leftarrow Z, W \leftarrow b\} = p(f(a,Z),f(Z,a))$
$p(X,Y) \{X \leftarrow f(Y), Y \leftarrow b\} = p(f(Y),b)$

Substitutions continued

Definition: Let $\theta = \{X_1 \leftarrow s_1, \ldots, X_m \leftarrow s_m\}$ and $\sigma = \{Y_1 \leftarrow t_1, \ldots, Y_n \leftarrow t_n\}$
The <u>composition</u> of $\theta\sigma$ is obtained from the set

$$\{X_1 \leftarrow s_1\sigma, \ldots, X_m \leftarrow s_m\sigma, Y_1 \leftarrow t_1, \ldots, Y_n \leftarrow t_n\}$$

by

- removing all $X_i \leftarrow s_i\sigma$ where $X_i = s_i\sigma$, and
- removing those $Y_j \leftarrow t_j$ for which Y_j is in $\{X_1, \ldots, X_m\}$.

<u>example</u>:

$\{X \leftarrow f(Z), Y \leftarrow W\}\{X \leftarrow a, Z \leftarrow a, W \leftarrow Y\} = \{X \leftarrow f(a), Z \leftarrow a, W \leftarrow Y\}$

<u>Properties</u>:

$E(\theta\sigma) = (E\,\theta)\sigma$

$(\theta\sigma)\omega = \theta(\sigma\omega)$

$\theta\varepsilon = \varepsilon\theta = \theta$

composition of substitutions is <u>not commutative</u>

$\{X \leftarrow f(Y)\}\{Y \leftarrow a\} = \{X \leftarrow f(a), Y \leftarrow a\}$

$\{Y \leftarrow a\}\{X \leftarrow f(Y)\} = \{Y \leftarrow a, X \leftarrow f(Y)\}$

Unifiers and Most General Unifiers (MGU)

Definition: Let s and t be two terms. A substitution θ is a unifier for s and t if $s\theta = t\theta$.

f(X,Y)
f(g(Z),Z)

unifier: $\theta = \{X \leftarrow g(Z), Y \leftarrow Z\}$
another unifier: $\sigma = \{X \leftarrow g(a), Y \leftarrow a, Z \leftarrow a\}$

Definition: A substitution θ is more general than substitution σ if there exists a substitution ω such that $\sigma = \theta\omega$

In previous example, $\sigma = \theta \{Z \leftarrow a\}$. θ is more general that σ

Definition: A unifier is said to be the most general unifier (mgu) of two terms if it is more general that any other unifier of the terms.

Algorithm for Most General Unifiers (MGU)

<u>Input:</u> S = $\{E_1, \ldots, E_n\}$
<u>Output</u>: mgu(S)
<u>Method</u>:

1. k := 0; $\sigma_k = \{\}$
2. If $S\sigma_k$ is a singleton, return σ_k otherwise find D_k, the disagreement set of $S\sigma_k$
3. If there exists a variable v and a term t in D_k such that <u>v does not appear in t</u>, then $\sigma_{k+1} = \sigma_k \{v \leftarrow t\}$, k++, goto step 2. Otherwise stop; S is not unifiable.

Note: The disagreement set of a set of terms is obtained by extracting the first sub-term in each term that is different in each.

S = { f(X,Y), f(g(Z),Z) }

k	σ_k	$S\sigma_k$	Disagreement set of $S\sigma_k$
0	{ }	{ f(X,Y), f(g(Z),Z) }	{ X, g(Z) }
1	{X ← g(Z) }	{ f(g(Z),Y), f(g(Z),Z) }	{ Y, Z }
2	{X ← g(Z), Y ← Z }	{ f(g(Z),Z) }	

Another MGU Example

S = { p(X,Y, X), p(f(Y), a, f(Z)) }

k	σ_k	$S\sigma_k$	Disagreement set of $S\sigma_k$
0	{ }	{ p(X,Y, X), p(f(Y), a, f(Z)) }	{ X, f(Y) }
1	{X ← f(Y) }	{ p(f(Y),Y, f(Y)), p(f(Y), a, f(Z)) }	{ Y, a }
2	{X ← f(a), Y ← a }	{ p(f(a),a, f(a)), p(f(a), a, f(Z)) }	{ Z, a}
3	{X ← f(a), Y ← a, Z ← a }	{ p(f(a),a, f(a)) }	

SLD Resolution
Informal Introduction

```
father(adam, mary).
newborn(mary).
proud(X) :- parent(X,), newborn(Y).
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).

?- proud(Z).
```

```
?- proud(Z).                    proud(X) :- parent(X,Y), newborn(Y).

?- parent(Z,Y), newborn(Y).     parent(X,Y) :- father(X,Y).

?- father(Z,Y), newborn(Y).     father(adam, mary).

?- newborn(mary).               newborn(mary).

■
```

SLD Resolution

Let **?- A_1, …, A_m.** be a goal/query (call it G_0) and let **B_0 :- B_1,…,B_n.**
be a renamed program fact/rule (call it R_0) whose head predicate unifies with A_1.
The variables in the program fact/rule are renamed so that there are
no common variables between the program fact/rule and G_0

Let mgu(A_1, B_0) = θ_1. The new goal/query (call it G_1) is

?- (B_1,…,B_n,A_2, …, A_m) θ_1

This process is repeated, i.e.

$$G_0 \xrightarrow[\theta_1]{R_0} G_1 \xrightarrow[\theta_2]{R_1} G_2 \ldots G_{(n-1)} \xrightarrow[\theta_n]{R_{(n-1)}} G_n \ldots$$

until (a) empty goal, or (b) cannot find program fact/rule to unify, or (c) infinite loop!
The sequence G_0, G_1,…, G_n,… is called a <u>SLD-derivation</u>. A finite sequence
ending in the empty goal is called a <u>SLD-Refutation</u>.

For a finite SLD-derivation, $\theta = \theta_1\theta_2\ldots\theta_n$ is called the <u>computed substitution</u>.

SLD Resolution continued

Example:

G_0: ?- proud(Z).
R_0: proud(X_0) :- parent(X_0,Y_0), newborn(Y_0).
$\theta_1 = \{X_0 \leftarrow Z\}$

G_1: ?- parent(Z,Y_0), newborn(Y_0).
R_1: parent(X_1,Y_1) :- father(X_1,Y_1).
$\theta_2 = \{X_1 \leftarrow Z, Y_1 \leftarrow Y_0\}$

G_2: ?- father(Z,Y_0), newborn(Y_0).
R_2: father(adam,mary).
$\theta_3 = \{Z \leftarrow adam, Y_0 \leftarrow mary\}$

G_3: ?- newborn(mary).
R_3: newborn(mary).
$\theta_4 = \{\}$

G_4: ■

$$\theta = \theta_1\theta_2\theta_3\theta_4$$
$$= \{X_0 \leftarrow Z\}\{X_1 \leftarrow Z, Y_1 \leftarrow Y_0\}\ \{Z \leftarrow adam, Y_0 \leftarrow mary\}\ \varepsilon$$
$$= \{X_0 \leftarrow adam, X_1 \leftarrow adam, Y_1 \leftarrow mary, Z \leftarrow adam, Y_0 \leftarrow mary\}$$

SLD Resolution - Another Example

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).

G_0: ?- grandfather(a,X).
R_0: grandfather(X_0,Z_0) :- father(X_0,Y_0), parent(Y_0,Z_0).
$\theta_1 = \{X_0 \leftarrow a,\ Z_0 \leftarrow X\}$

G_1: ?- father(a,Y_0), parent(Y_0,X).
R_1: father(a,b).
$\theta_2 = \{Y_0 \leftarrow b\}$

G_2: ?- parent(b,X).
R_2: parent(X_2,Y_2) :- mother(X_2,Y_2).
$\theta_3 = \{X_2 \leftarrow b,\ Y_2 \leftarrow X\}$

G_3: ?- mother(b,X).
R_3: mother(b,c).
$\theta_4 = \{X \leftarrow c\}$

G_4: ■

$\theta = \theta_1\theta_2\theta_3\theta_4$
$= \{X_0 \leftarrow a,\ Z_0 \leftarrow X\}\ \{Y_0 \leftarrow b\}\{X_2 \leftarrow b,\ Y_2 \leftarrow X\}\{X \leftarrow c\}$
$= \{X_0 \leftarrow a,\ Y_0 \leftarrow b,\ X_2 \leftarrow b,\ Y_2 \leftarrow c,\ X \leftarrow c\}$

SLD Refutation Tree

You may have noticed that in SLD Resolution there may be <u>multiple choices</u> for the program fact/rule. In fact, Prolog implementation will try these choices out <u>exhaustively</u> (in a <u>depth-first</u> manner using <u>back-tracking</u>) to obtain one or more SLD-refutations, resulting in one or more answers to the original goal/query.

Example:

G_0: ?- grandfather(a,X).
R_0: grandfather(X_0,Z_0) :- father(X_0,Y_0), parent(Y_0,Z_0).
$\theta_1 = \{X_0 \leftarrow a, Z_0 \leftarrow X\}$

G_1: ?- father(a,Y_0), parent(Y_0,X).
R_1: father(a,b).
$\theta_2 = \{Y_0 \leftarrow b\}$

G_2: ?- parent(b,X).
R_2: parent(X_2,Y_2) :- father(X_2,Y_2).
$\theta_3 = \{X_2 \leftarrow b, Y_2 \leftarrow X\}$

G_3: ?- father(b,X).
STUCK!! cannot progress; Prolog will backtrack and try other possibilities.

SLD Refutation Tree

?- grandfather(a,X).

grandfather(X_0,Z_0) :- father(X_0,Y_0), parent(Y_0,Z_0).
$\theta_1 = \{X_0 \leftarrow a,\ Z_0 \leftarrow X\}$

?- father(a,Y0), parent(Y0,X).

father(a,b).
$\theta_2 = \{Y_0 \leftarrow b\}$

?- parent(b,X).

parent(X_2,Y_2) :- father(X_2,Y_2).
$\theta_3 = \{X_2 \leftarrow b,\ Y_2 \leftarrow X\}$

?- father(b,X).

FAIL

parent(X_2,Y_2) :- mother(X_2,Y_2).
$\theta_3 = \{X_2 \leftarrow b,\ Y_2 \leftarrow X\}$

?- mother(b,X).

mother(b,c).
$\theta_4 = \{X \leftarrow c\}$

■

$$\theta = \theta_1\theta_2\theta_3\theta_4$$

$= \{X_0 \leftarrow a,\ Z_0 \leftarrow X\}\ \{Y_0 \leftarrow b\}\{X_2 \leftarrow b,\ Y_2 \leftarrow X\}\{X \leftarrow c\}$

$= \{X_0 \leftarrow a,\ Y_0 \leftarrow b,\ X_2 \leftarrow b,\ Y_2 \leftarrow c,\ X \leftarrow c\}$

Practice Problems

1. Find the mgu, if any, for the following sets:

S = { p(X,f(X)), p(Y,f(a)) }

T = { p(a,X), p(X,f(X)) }

2. Consider the following program:

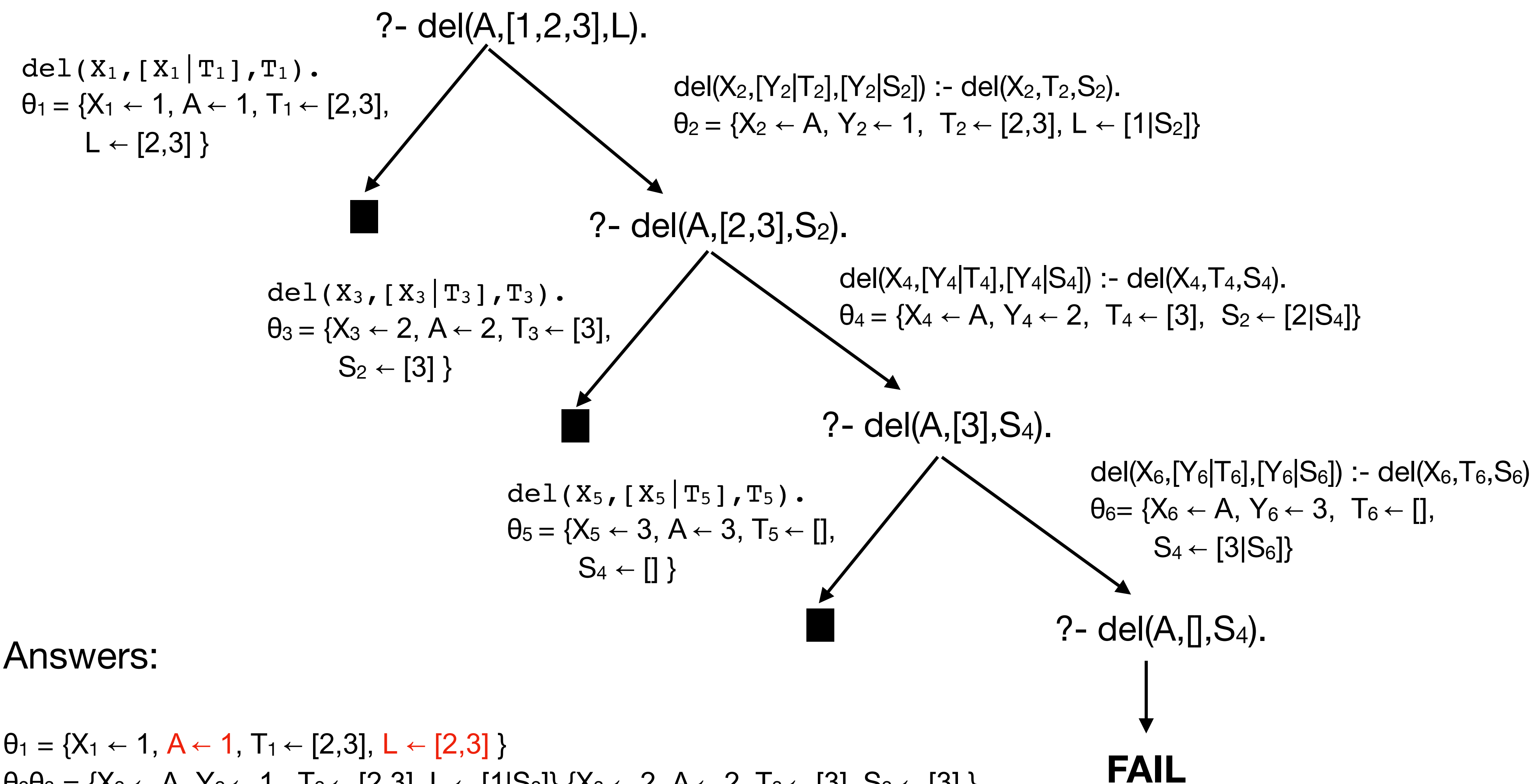
```
p(Y) :- q(X,Y), r(Y).
p(X) :- q(X,X).
q(X,X) :- s(X).
r(b).
s(a).
s(b).
```

Draw the complete SLD-refutation tree for the goal:

```
?- p(X).
```

SLD Refutation Tree - Another Example

```
del(X,[X|T],T).
del(X,[Y|T],[Y|S]) :- del(X,T,S).
```



Answers:

$\theta_1 = \{X_1 \leftarrow 1,\ A \leftarrow 1,\ T_1 \leftarrow [2,3],\ L \leftarrow [2,3]\ \}$

$\theta_2\theta_3 = \{X_2 \leftarrow A,\ Y_2 \leftarrow 1,\ T_2 \leftarrow [2,3],\ L \leftarrow [1|S_2]\}\ \{X_3 \leftarrow 2,\ A \leftarrow 2,\ T_3 \leftarrow [3],\ S_2 \leftarrow [3]\ \}$

$\quad = \{X_2 \leftarrow 2,\ Y_2 \leftarrow 1,\ T_2 \leftarrow [2,3],\ L \leftarrow [1,3],\ X_3 \leftarrow 2,\ A \leftarrow 2,\ T_3 \leftarrow [3],\ S_2 \leftarrow [3]\}$

$\theta_2\theta_4\theta_5 = \{X_2 \leftarrow A,\ Y_2 \leftarrow 1,\ T_2 \leftarrow [2,3],\ L \leftarrow [1|S_2]\}\ \{X_4 \leftarrow A,\ Y_4 \leftarrow 2,\ T_4 \leftarrow [3],\ S_2 \leftarrow [2|S_4]\}\ \{X_5 \leftarrow 3,\ A \leftarrow 3,\ T_5 \leftarrow [],\ S_4 \leftarrow []\ \}$

$\quad = \{X_2 \leftarrow 3,\ Y_2 \leftarrow 1,\ T_2 \leftarrow [2,3],\ L \leftarrow [1,2],\ X_4 \leftarrow 3,\ Y_4 \leftarrow 2,\ T_4 \leftarrow [3],\ S_2 \leftarrow [2],\ X_5 \leftarrow 3,\ A \leftarrow 3,\ T_5 \leftarrow [],\ S_4 \leftarrow []\}$