Refutation in Predicate Logic

\[
\begin{align*}
\text{parent}(pam, bob). & \quad \text{anc}(X,Y) : - \\
\text{parent}(tom, bob). & \quad \text{parent}(X,Y).
\end{align*}
\]

- For what values of \( Q \) is \( \text{anc}(\text{tom},Q) \) a logical consequence of the above program?
- Negate the goal \( F \): i.e. \( \forall Q. \neg \text{anc}(\text{tom},Q) \).
- 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:
    \[
    \forall A, B, C. (p(A,B) \lor \neg q(A,C) \lor r(B,C))
    \]
  - 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.

Refutation: An Example

\[
\begin{align*}
\text{parent}(pam, bob). & \quad \text{anc}(X,Y) : - \\
\text{parent}(tom, bob). & \quad \text{parent}(X,Y).
\end{align*}
\]

\[
\begin{align*}
\leftarrow \text{anc}(\text{tom},Q) & \quad \leftarrow \text{parent}(\text{tom},Q) \\
\text{anc}(X,Y) & \quad \leftarrow \text{parent}(X,Y) \\
\text{parent}(\text{tom}, \text{bob}) & \quad \Box \quad Q=\text{bob}
\end{align*}
\]
Refutation: An Example (contd.)

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

\[ \text{anc}(X,Y) \leftarrow \text{parent}(X,Y). \]
\[ \text{anc}(X,Y) \leftarrow \text{parent}(X,Z), \text{anc}(Z,Y). \]

\[ \text{anc}(X,Y) \leftarrow \text{parent}(tom,Z’), \text{anc}(Z’,Q). \]

\[ \text{anc}(X,Y) \leftarrow \text{parent}(bob,Q). \]

\[ \text{anc}(X,Y) \leftarrow \text{parent}(bob,Q). \]
\[ \text{anc}(X,Y) \leftarrow \text{parent}(bob,ann) \leftarrow \]
\[ Q=\text{ann} \]

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.
  - \( X \) and \( f(X) \) do not unify (but they “match” in Prolog!)
Substitutions

A substitution is a mapping between variables and values (terms).

- Denoted by \( \{X_1/t_1, X_2/t_2, \ldots, X_n/t_n\} \) such that
  - \( X_i \neq t_i \), and
  - \( X_i \) and \( X_j \) are distinct variables when \( i \neq j \).
- Empty substitution is denoted by \( \epsilon \).
- A substitution is said to be a **renaming** if it is of the form
  \( \{X_1/Y_1, \ldots, X_n/Y_n\} \) and \( Y_1, \ldots, Y_n \) is a permutation of \( X_1, \ldots, X_n \).
- Example: \( \{X/Y, Y/X\} \) is a renaming substitution.

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 \( \{X_1/t_1, \ldots, X_n/t_n\} \) to a term/formula \( F \):
  - is a term/formula obtained by simultaneously replacing every **free** occurrence of \( X_i \) in \( F \) by \( t_i \).
  - Denoted by \( F\theta \) [and \( F\theta \) is said to be an instance of \( F \)]
- Example:
  \[ p(f(X, Z), f(Y, a))\{X/g(Y), Y/Z, Z/a\} = p(f(g(Y), a), f(Z, a)) \]
Composition of Substitutions

- Composition of substitutions $\theta = \{X_1/s_1, \ldots, X_m/s_m\}$ and $\sigma = \{Y_1/t_1, \ldots, Y_n/t_n\}$:
  - First form the set $\{X_1/s_1\sigma, \ldots, X_m/s_m\sigma, Y_1/t_1, \ldots, Y_n/t_n\}$
  - Remove from the set $X_i/s_i\sigma$ if $s_i\sigma = X_i$
  - Remove from the set $Y_j/t_j$ if $Y_j$ is identical to some variable $X_i$
- 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 $\sigma = \{Y/a\}$
  - $\theta\sigma = \{X/f(a), Y/a\}$
  - $\theta\sigma = \{X/f(Y), Y/a\}$
- Composition is not commutative but is associative: $\theta(\sigma\gamma) = (\theta\sigma)\gamma$
- Also, $E(\theta\sigma) = (E\theta)\sigma$

Idempotence

- A substitution $\theta$ 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 $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$,
  - $\text{Dom}(\theta) = \{X_1, X_2, \ldots X_n\}$
  - $\text{Range}(\theta) = \text{set of all variables in } t_1, \ldots t_n$
- A substitution $\theta$ is idempotent iff $\text{Dom}(\theta) \cap \text{Range}(\theta) = \emptyset$
Unification

A substitution \( \theta \) is a **unifier** of two terms \( s \) and \( t \) if \( s\theta \) is identical to \( t\theta \).

\( \theta \) is a unifier of a set of equations \( \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\} \), if for all \( i \), \( s_i\theta = t_i\theta \).

A substitution \( \theta \) is more general than \( \sigma \) (written as \( \theta \succeq \sigma \)) if there is a substitution \( \omega \) such that \( \sigma = \theta\omega \).

A substitution \( \theta \) is a **most general unifier** (mgu) of two terms (or a set of equations) if for every unifier \( \sigma \) of the two terms (or equations) \( \theta \succeq \sigma \).

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

Equations and Unifiers

A set of equations \( \mathcal{E} \) is in **solved form** if it is of the form \( \{X_1 \doteq t_1, \ldots, X_n \doteq t_n\} \) iff no \( X_i \) appears in any \( t_j \).

Given a set of equations \( \mathcal{E} = \{X_1 \doteq t_1, \ldots, X_n \doteq t_n\} \) the substitution \( \{X_1/t_1, \ldots X_n/t_n\} \) is an idempotent mgu of \( \mathcal{E} \).

Two sets of equations \( \mathcal{E}_1 \) and \( \mathcal{E}_2 \) 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 \( \{s \doteq t\} \).

If there is no equivalent solved form, there is no mgu.
A Simple Unification Algorithm (via Examples)

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

\[
\{f(X, g(Y)) \doteq f(g(Z), Z)\} \Rightarrow \{X \doteq g(Z), g(Y) \doteq Z\} \\
\Rightarrow \{X \doteq g(Z), Z \doteq g(Y)\} \\
\Rightarrow \{X \doteq g(g(Y)), Z \doteq g(Y)\}
\]

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

\[
\{f(X, g(X), b) \doteq f(a, g(Z), Z)\} \Rightarrow \{X \doteq a, g(X) \doteq g(Z), b \doteq Z\} \\
\Rightarrow \{X \doteq a, a \doteq Z, b \doteq Z\} \\
\Rightarrow \{X \doteq a, Z \doteq a, b \doteq Z\} \\
\Rightarrow \{X \doteq a, Z \doteq a, b \doteq a\} \\
\Rightarrow \text{fail}
\]

A Simple Unification Algorithm

Given a set of equations $E$:

repeat

select $s \doteq t \in E$;

\textbf{case} $s \doteq t$ \textbf{of}

1. $f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)$:
   replace the equation by $s_i \doteq t_i$ for all $i$

2. $f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_m)$, $f \neq g$ or $n \neq m$:
   halt with \textbf{failure}

3. $X \doteq X$: remove the equation

4. $t \doteq X$: where $t$ is not a variable
   replace equation by $X \doteq t$

5. $X \doteq t$: where $X \neq t$ and $X$ occurs more than once in $E$:
   \textbf{if} $X$ is a proper subterm of $t$
   \textbf{then} halt with \textbf{failure} \hspace{1cm} (5a)
   \textbf{else} replace all other $X$ in $E$ by $t$ \hspace{1cm} (5b)

until no action is possible for any equation in $E$

return $E$
A Simple Unification Algorithm (More Examples)

- Example 1: Find the mgu of \( f(X, g(Y)) \) and \( f(g(Z), Z) \)
  \[
  \{f(X, g(Y)) \equiv f(g(Z), Z)\} \Rightarrow \{X \equiv g(Z), g(Y) \equiv Z\} \quad \text{case 1}
  \]
  \[
  \Rightarrow \{X \equiv g(Z), Z \equiv g(Y)\} \quad \text{case 4}
  \]
  \[
  \Rightarrow \{X \equiv g(g(Y)), Z \equiv g(Y)\} \quad \text{case 5b}
  \]

- Example 3: Find the mgu of \( f(X, g(X)) \) and \( f(Z, Z) \)
  \[
  \{f(X, g(X)) \equiv f(Z, Z)\} \Rightarrow \{X \equiv Z, g(X) \equiv Z\} \quad \text{case 1}
  \]
  \[
  \Rightarrow \{X \equiv Z, g(Z) \equiv Z\} \quad \text{case 5b}
  \]
  \[
  \Rightarrow \{X \equiv Z, Z \equiv g(Z)\} \quad \text{case 4}
  \]
  \[
  \Rightarrow \text{fail} \quad \text{case 5a}
  \]

Complexity of the unification algorithm

Consider \( \mathcal{E} = \{g(X_1, \ldots, X_n) \equiv g(f(X_0, X_0), f(X_1, X_1), \ldots, f(X_{n-1}, X_{n-1}))\} \).

- By applying case 1 of the algorithm, we get
  \[
  \{X_1 = f(X_0, X_0), X_2 = f(X_1, X_1), \ldots, X_n = f(X_{n-1}, X_{n-1})\}
  \]

- If terms are kept as trees, the final value for \( X_n \) 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.
**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 $\theta$ is an mgu of $s$ and $t$, and $\omega$ is a renaming, then $\theta\omega$ is an mgu of $s$ and $t$.
- If $\theta$ and $\sigma$ are mgus of $s$ and $t$, then there is a renaming $\omega$ such that $\theta = \sigma\omega$.
  - mgu is unique up to renaming.

**SLD Resolution**

\[
\leftarrow A_1, \ldots, A_{i-1}, \underline{A_i}, A_{i+1}, \ldots, A_m \quad B_0 \leftarrow B_1, \ldots, B_n \\
\leftarrow (A_1, \ldots, A_{i-1}, B_1, \ldots, B_n, A_{i+1}, \ldots, A_m)\theta
\]

where:
- $A_j$s are atomic formulas
- $B_0 \leftarrow B_1, \ldots, B_n$ is a (renamed) definite clause in the program
- $\theta = \text{mgu}(A_i, B_0)$
- $A_i$ is called the selected atom.
- Given a goal $\leftarrow A_1, \ldots, A_n$ a function called the selection function or computation rule selects $A_i$. 
SLD Resolution

SLD resolution (contd.)

- 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 \( C \):

\[
G \xrightarrow{C} G'
\]

- An **SLD Derivation** is a sequence

\[
G_0 \xrightarrow{C_0} G_1 \cdots G_i \xrightarrow{C_i} G_{i+1} \cdots
\]

---

Refutation & SLD Derivation

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

\[
\text{proj}(X, Y) :- \\
\quad \text{parent}(X, Y).
\]

\[
\text{anc}(X, Y) :- \\
\quad \text{parent}(X, Z), \\
\quad \text{anc}(Z, Y).
\]

\[
\begin{align*}
\text{anc}(X, Y) & \leftarrow \text{parent}(X, Y) \\
\text{anc}(tom, Q) & \leftarrow \text{parent}(tom, Q) \\
\text{anc}(tom, Q) & \leftarrow □
\end{align*}
\]

\[
\begin{align*}
\text{parent}(tom, Q) & \leftarrow □
\end{align*}
\]
SLD Derivation (contd.)

\[ \text{parent}(pam, \text{bob}). \quad \text{parent}(\text{tom}, \text{bob}). \quad \text{parent}(\text{tom}, \text{liz}). \quad \text{parent}(\text{bob}, \text{ann}). \quad \text{parent}(\text{bob}, \text{pat}). \quad \text{parent}(\text{pat}, \text{jim}). \]

\[ \leftarrow \text{anc}(\text{tom}, Q) \]

\[ \text{anc}(X, Y) \leftarrow \text{parent}(X, Z), \text{anc}(Z, Y) \]

\[ \leftarrow \text{parent}(\text{tom}, Z'), \text{anc}(Z', Q) \]

\[ \quad \text{parent}(\text{tom}, \text{bob}) \leftarrow \]

\[ \leftarrow \text{anc}(\text{bob}, Q) \]

\[ \text{anc}(X, Y) \leftarrow \text{parent}(X, Y) \]

\[ \leftarrow \text{parent}(\text{bob}, Q) \]

\[ \quad \text{parent}(\text{bob}, \text{ann}) \leftarrow \]

\[ \square \]

\[ Q = \text{ann} \]

---

**Computed Answer Substitution**

- Let \( \theta_0, \theta_1, \ldots, \theta_{n-1} \) be the sequence of mgu used in derivation

\[ G_0 \overset{C_0}{\rightsquigarrow} G_1 \cdots G_{n-1} \overset{C_{n-1}}{\rightsquigarrow} G_n \]

Then \( \theta = \theta_0 \theta_1 \cdots \theta_{n-1} \) is the computed substitution of the derivation.

- Example:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Clause Used</th>
<th>mgu</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{anc}(\text{tom}, Q) )</td>
<td>( \text{anc}(X', Y') :)</td>
<td>( \theta_0 = {X'/\text{tom}, Y'/Q} )</td>
</tr>
<tr>
<td>( \text{parent}(\text{tom}, Z'), \text{anc}(Z', Q) )</td>
<td>( \text{parent}(\text{tom}, \text{bob}). \quad \text{anc}(X'', Y'') :)</td>
<td>( \theta_1 = {Z'/\text{bob}} )</td>
</tr>
<tr>
<td>( \text{parent}(\text{bob}, Q) )</td>
<td>( \text{parent}(\text{bob}, \text{ann}). )</td>
<td>( \theta_2 = {X''/\text{bob}, Y''/Q} )</td>
</tr>
<tr>
<td>( \square )</td>
<td></td>
<td>( \theta_3 = {Q/\text{ann}} )</td>
</tr>
</tbody>
</table>

- Computed substitution for the above derivation is \( \theta_0 \theta_1 \theta_2 \theta_3 = \{X'/\text{tom}, Y'/\text{ann}, Z'/\text{bob}, X''/\text{bob}, Y''/\text{ann}, Q/\text{ann}\} \)
**Computed Answer Substitution (contd.)**

- 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 \( G_n = \square \) (i.e., an empty goal) is an **SLD refutation** of \( G_0 \).

- 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

  \[ \{ X'/\text{tom}, Y'/\text{ann}, Z'/\text{bob}, X''/\text{bob}, Y''/\text{ann}, Q/\text{ann} \} \]

  restricted to \( Q \):

  \[ \{ Q/\text{ann} \} \]

**Failed SLD Derivation**

- A derivation of a goal clause \( G_0 \) whose last element is not empty, and cannot be resolved with any clause of the program.

- Example: consider the following program:

  \[
  \begin{align*}
  \text{grandfather}(X,Z) & : - \text{father}(X,Y), \text{parent}(Y,Z). \\
  \text{parent}(X,Y) & : - \text{father}(X,Y). \\
  \text{parent}(X,Y) & : - \text{mother}(X,Y). \\
  \text{father}(a,b). \\
  \text{mother}(b,c).
  \end{align*}
  \]

  A derivation of \( \text{grandfather}(a,Q) \) is:

  \[
  \begin{align*}
  \leadsto & \text{father}(a,Y'), \text{parent}(Y',Q) \\
  \leadsto & \text{parent}(b,Q) \\
  \leadsto & \text{father}(b,Q)
  \end{align*}
  \]
SLD Tree

A tree where every path is an SLD derivation

\[
\text{grandfather}(X,Z) :- \\
\quad \text{father}(X,Y), \text{parent}(Y,Z).
\]

\[
\text{parent}(X,Y) :- \text{father}(X,Y).
\]

\[
\text{parent}(X,Y) :- \text{mother}(X,Y).
\]

\[
\text{father}(a,b).
\]

\[
\text{mother}(b,c).
\]

\[
\text{← grandfather}(a, Q)
\]

\[
\quad \text{← father}(a, Z'), \text{parent}(Z', Q)
\]

\[
\quad \text{← parent}(b, Q)
\]

\[
\quad \text{← father}(b, Q)
\]

\[
\quad \text{← mother}(b, Q)
\]

Soundness of SLD resolution

- Let $P$ be a definite program, $\mathcal{R}$ be a computation rule, and $\theta$ 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 $\theta$.
    Then $(\forall F) \models (\forall F')$. 