Foundations of Programming Languages and Software Engineering

Universität Freiburg

July 2011

Unification
(Syntactic) Unification

Definition

(Syntactic) unification is the following problem: Given \( s \) and \( t \), find a substitution \( \sigma \) such that \( \sigma s = \sigma t \).

If \( \sigma s = \sigma t \), then \( \sigma \) is called a unifier of \( s \) and \( t \) or a solution to the equation \( s \equiv t \).
(Syntactic) Unification

Definition

(Syntactic) unification is the following problem: Given $s$ and $t$, find a substitution $\sigma$ such that $\sigma s = \sigma t$.

If $\sigma s = \sigma t$, then $\sigma$ is called a unifier of $s$ and $t$ or a solution to the equation $s \equiv t$.

More generally, unification is about deciding $\sigma s \approx \varepsilon \sigma t$ (non-ground word problem). But here, by unification we mean syntactic unification.
(Syntactic) Unification

Definition

(Syntactic) unification is the following problem: Given $s$ and $t$, find a substitution $\sigma$ such that $\sigma s = \sigma t$.

If $\sigma s = \sigma t$, then $\sigma$ is called a unifier of $s$ and $t$ or a solution to the equation $s \equiv t$.

More generally, unification is about deciding $\sigma s \approx \epsilon \sigma t$ (non-ground word problem). But here, by unification we mean syntactic unification.

Unification is decidable.

Unification is theoretically and practically interesting:
- Symbolic computation algorithms
- Prolog
- Type inference
### Unifiers

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = f(a)$</td>
<td>has exactly one unifier: ${x \mapsto a}$</td>
</tr>
<tr>
<td>$x = f(y)$</td>
<td>has many unifiers:</td>
</tr>
<tr>
<td>$n$</td>
<td>${x \mapsto f(y)}$, ${x \mapsto f(a), y \mapsto a}$, ...</td>
</tr>
<tr>
<td>$f(x) = g(y)$</td>
<td>has no unifier</td>
</tr>
<tr>
<td>$x = f(x)$</td>
<td>has no unifier</td>
</tr>
</tbody>
</table>

- An equation $s = t$ may have zero, one, or more solutions.
- Some solutions are **more general** than others:  
  $\{x \mapsto f(y)\}$ is more general than $\{x \mapsto f(a), y \mapsto a\}$. 
A unification problem is a finite set of equations $S = \{s_1 =? t_1, \ldots, s_n =? t_n\}$.

A unifier or solution of $S$ is a substitution $\sigma$ such that $\sigma s_i = \sigma t_i$ for all $i = 1, \ldots, n$.

$\mathcal{U}(S)$ denotes the set of all unifiers of $S$.

$S$ is unifiable if $\mathcal{U}(S) \neq \emptyset$. 
Definition

A unification problem \( S = \{ x_1 = ? t_1, \ldots, x_n = ? t_n \} \) is in solved form iff

- the \( x_i \) are pairwise distinct variables,
- none of the \( x_i \) occurs in any of the \( t_j \).

In this case, we define the substitution \( \vec{S} \) as follows:

\[
\vec{S} := \{ x_1 \mapsto t_1, \ldots, x_n \mapsto t_n \}
\]

- It is easy to see that \( \vec{S} \) is a unifier of \( S \).
- Next we show how to transform a unification problem into solved form, provided the unification problem has a solution.
## Transformation Rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DELETE</strong></td>
<td>${ t = ? t } \uplus S \implies S$</td>
</tr>
<tr>
<td><strong>DECOMPOSE</strong></td>
<td>${ f(t_n) = ? f(u_n) } \uplus S \implies { t_1 = ? u_1, \ldots, t_n = ? u_n } \uplus S$</td>
</tr>
<tr>
<td><strong>ORIENT</strong></td>
<td>${ t = ? x } \uplus S \implies { x = ? t } \uplus S$ if $t \not\in X$</td>
</tr>
<tr>
<td><strong>ELIMINATE</strong></td>
<td>${ x = ? t } \uplus S \implies { x = ? t } \uplus { x \mapsto t }(S)$ if $x \in \text{Var}(S)$ and $x \not\in \text{Var}(t)$ (“occurs check”)</td>
</tr>
</tbody>
</table>

- The symbol $\uplus$ denotes disjoint union: $M_1 \uplus M_2 := M_1 \cup M_2$ provided $M_1 \cap M_2 = \emptyset$.
- Applying a substitution to a set of equations $S$ means applying it to both sides of all equations in $S$. 
Example (1)

Success

\{ x =? f(a), g(x, x) =? g(x, y) \} \Rightarrow \text{ELIMINATE}
\{ x =? f(a), g(f(a), f(a)) =? g(f(a), y) \} \Rightarrow \text{DECOMPOSE}
\{ x =? f(a), f(a) =? f(a), f(a) =? y \} \Rightarrow \text{DELETE}
\{ x =? f(a), f(a) =? y \} \Rightarrow \text{ORIENT}
\{ x =? f(a), y =? f(a) \}
Example (2)

Failure

\[
\begin{align*}
\{ f(x, x) =? f(y, g(y)) \} & \implies \text{DECOMPOSE} \\
\{ x =? y, x =? g(y) \} & \implies \text{ELIMINATE} \\
\{ x =? y, y =? g(y) \} & \implies \text{ELIMINATE}
\end{align*}
\]

- No transformation rule is applicable to \( \{ x =? y, y =? g(y) \} \).
- \text{ELIMINATE} is not applicable to \( y =? g(y) \) because the occurs check fails.
Definition

\[ \text{Unify}(S) = \text{while there is some } T \text{ such that } S \rightarrow T \text{ do} \]
\[ S := T; \]
\[ \text{end while} \]
\[ \text{if } S \text{ is in solved form then return } \dot{S} \text{ else fail} \]
**Properties of Unify**

- *Unify* is nondeterministic:
  If more than one transformation rule is applicable, say $S \Rightarrow T_1$ and $S \Rightarrow T_2$, then *Unify* may choose arbitrarily between $T_1$ and $T_2$.

- *Unify* is sound:
  If *Unify*(S) returns a substitution $\sigma$, then $\sigma$ is a unifier of $S$.

- *Unify* is complete:
  If a unification problem $S$ is solvable then *Unify*(S) does not fail.

- *Unify* terminates for all inputs.
Further Properties of *Unify*

*Unify* has some further properties that we can only state loosely because we did not formally introduce the necessary concepts:

- *Unify* computes a **most general** unifier: e.g., for $x = ? f(y)$ it will compute $\{x \mapsto f(y)\}$, not $\{x \mapsto f(a), y \mapsto a\}$.

- *Unify* computes an **idempotent** unifier, i.e., a unifier $\sigma$ such that $\sigma\sigma = \sigma$. This rules out strange solutions such as $\{x \mapsto f(y), z_1 \mapsto z_2, z_2 \mapsto z_1\}$ for the problem $x = ? f(y)$. 
Earlier Failure Detection

- Detecting unsolvability can be expensive because *Unify* first computes a normal form.
- But if the unification problem contains
  - an equation $f(\ldots) =^? g(\ldots)$ with $f \neq g$ or
  - an equation $x =^? t$ with $x \in \text{Var}(t)$ and $x \neq t$
  then failure is immediate.
- Introduce a special unification problem $\perp$ which is not in solved form.
- Add two more transformation rules:

  - **C\text{LASH}** \[
    \{ f(t_n) =^? g(u_n) \} \uplus S \implies \perp \quad \text{if } f \neq g
  \]
  - **O\text{CCURS–C\text{HECK}}** \[
    \{ x =^? t \} \uplus S \implies \perp \quad \text{if } x \in \text{Var}(t) \text{ and } x \neq t
  \]