# 16 Foundation of Programming Languages and Software Engineering: *Satisfiability*

Summer Term 2010

Robert Elsässer

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

# Central Problems of Equational Reasoning

**Definition (Validity)**

$s \approx t$ is valid in $\mathcal{E}$ iff $s \approx_{\mathcal{E}} t$

**Definition (Satisfiability)**

$s \approx t$ is satisfiable in $\mathcal{E}$ if there exists a substitution $\sigma$ such that $\sigma s \approx_{\mathcal{E}} \sigma t$.

# Unification

## Definition

Unification is the process of solving the satisfiability problem: Given $\mathcal{E}$ and $s$ and $t$, find a substitution $\sigma$ such that $\sigma s \approx_{\mathcal{E}} \sigma t$.

- If $s$ and $t$ are ground terms, unification degenerates to the ground word problem.
- The ground word problem is undecidable, and so is the unification problem.

# Syntactic Unification

## Definition

- Syntactic unification is the unification problem restricted to the empty set of identities ($\mathcal{E} = \emptyset$).
- 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 $s =^? t$.

- Syntactic unification is decidable.
- Syntactic unification is theoretically and practically interesting:
  - Symbolic computation algorithms
  - Interpreters for Prolog
  - Type inference

# Example

## Unifiers

$$\begin{array}{rl} f(x) =^? f(a) & \text{has exactly one unifier: } \{x \mapsto a\} \\ x =^? f(y) & \text{has many unifiers:} \\ & \{x \mapsto f(y)\}, \{x \mapsto f(a), y \mapsto a\}, \ldots \\ f(x) =^? g(y) & \text{has no unifier} \\ x =^? f(x) & \text{has no unifier} \end{array}$$

- 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 $\{x \mapsto f(a), y \mapsto a\}$.

# What is a more general substitution?

## Definition

- A substitution $\sigma$ is more general than a substitution $\sigma'$ iff there is a substitution $\delta$ such that $\sigma' = \delta\sigma$
- We write $\sigma \lesssim \sigma'$ if $\sigma$ is more general than $\sigma'$.
- If $\sigma \lesssim \sigma'$ then $\sigma'$ is called an instance of $\sigma$.

## Example

- Suppose $\sigma = \{x \mapsto f(y)\}$ and $\sigma' = \{x \mapsto f(a), y \mapsto a\}$.
- Define $\delta = \{y \mapsto a\}$.
- Then $\sigma' = \delta\sigma$, hence $\sigma \lesssim \sigma'$.

## Lemma

The relation $\lesssim$ is reflexive and transitive.

# Unification Problems

## Definition

- 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 $i = 1, ..., n$.
- $\mathcal{U}(S)$ denotes the set of all unifiers of $S$.
- $S$ is unifiable if $\mathcal{U}(S) \neq \emptyset$
- A substitution $\sigma$ is a most general unifier (mgu) of $S$ if $\sigma$ is a least element of $\mathcal{U}(S)$:
  - $\sigma \in \mathcal{U}(S)$ and
  - for all $\sigma' \in \mathcal{U}(S)$: $\sigma \lesssim \sigma'$

# Example

- Suppose $S = \{x =^? y\}$.
- $\sigma := \{x \mapsto y\}$ is an mgu of $S$:
  - Suppose $\theta$ also unifies $S$. Then
  - $\theta(x) = \theta(y) = \theta\sigma(x)$ and
  - $\theta(z) = \theta\sigma(z)$ for any other variable $z$ (including $z = y$).
- $\sigma' := \{y \mapsto x\}$ is also an mgu of $S$.
- $\tau := \{x \mapsto z, y \mapsto z\}$ is a unifier but not an mgu of $S$ because $\tau \not\lesssim \sigma$:
  - Consider $\delta := \{z \mapsto y\}$.
  - Then $\delta\tau = \{x \mapsto y, z \mapsto y\} \neq \sigma$.
- $\sigma'' := \{x \mapsto y, z_1 \mapsto z_2, z_2 \mapsto z_1\}$ is an mgu of $S$:
  - $\sigma'' \lesssim \sigma$ because $\sigma = \{z_1 \mapsto z_2, z_2 \mapsto z_1\}\sigma''$
  - If $\theta$ is a unifier of $S$ then $\sigma \lesssim \theta$, hence $\sigma'' \lesssim \theta$.

# Idempotent Substitutions

- Our algorithm for finding mgus should not return a solution such as $\sigma'' = \{x \mapsto y, z_1 \mapsto z_2, z_2 \mapsto z_1\}$.
- Note that we have $\sigma''\sigma'' = \{x \mapsto y\} \neq \sigma''$

**Definition**

A substitution $\sigma$ is idempotent iff $\sigma = \sigma\sigma$.

**Lemma**

A substitution $\sigma$ is idempotent iff $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) = \emptyset$.

(For a substitution $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, $\mathcal{VR}an(\sigma)$ denotes the set of variables occurring in $t_1, \ldots, t_n$.)

# Proof

"$\Rightarrow$" Given $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ with $x_i \neq t_i$ for all $i$. Suppose $\sigma$ is idempotent and $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) \neq \emptyset$.

- Then there exists $i, j \in \{1, \ldots, n\}$ such that $x_i$ occurs in $t_j$.
- By idempotency, $\sigma(t_j) = \sigma(\sigma(x_j)) = \sigma(x_j) = t_j$.
- But then $t_i = \sigma(x_i) = x_i$ which is a contradiction.
- Hence $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) = \emptyset$.

"$\Leftarrow$" Suppose $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) = \emptyset$ and let $t$ be an arbitrary term. We prove $\sigma(t) = \sigma\sigma(t)$ by term induction.

- $t = x$. If $x \in \mathcal{D}om(\sigma)$, then $\sigma(x)$ does not contain any variables from $\mathcal{D}om(\sigma)$, so $\sigma\sigma(x) = \sigma(x)$.
  If $x \notin \mathcal{D}om(\sigma)$, then $\sigma(x) = x$, hence $\sigma\sigma(x) = \sigma(x)$.
- $t = f$. Then $f = \sigma(f) = \sigma\sigma(f)$.
- $t = f(t_1, \ldots, t_n)$: Then $\sigma\sigma(t) = f(\sigma\sigma(t_1), \ldots, \sigma\sigma(t_n)) \stackrel{IH}{=} f(\sigma(t_1), \ldots, \sigma(t_n)) = \sigma(t)$

# Idempotent mgus

- An mgu is not necessarily idempotent (as seen before).
- But . . .

**Theorem**

If a unification problem $S$ has a solution, then it has an idempotent mgu.

Next step: Develop an algorithm that computes an idempotent mgu for a given unification problem or fails if there is no solution.

# Solving the Unification Problem

## 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:

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

- We now show that $\vec{S}$ is an idempotent mgu of $S$.
- Then we show how to transform a unification problem into solved form, provided the unification problem has a solution.

# Solved Forms are mgus (1)

**Lemma**

If $S$ is in solved form then $\sigma = \sigma\vec{S}$ for all $\sigma \in \mathcal{U}(S)$.

*Proof.* Let $S = \{x_1 =^? t_1, \ldots, x_n =^? t_n\}$. We show by case distinction that $\sigma(x) = \sigma\vec{S}(x)$ for all variables $x$.

1. $x \in \{x_1, \ldots, x_n\}$, e.g. $x = x_k$ :
   Then we have $\sigma(x) \overset{\sigma \in \mathcal{U}(S)}{=} \sigma(t_k) = \sigma\vec{S}(x)$.
2. $x \notin \{x_1, \ldots, x_n\}$.
   Then $\sigma(x) = \sigma\vec{S}(x)$ because $\vec{S}(x) = x$.

# Solved Forms are mgus (2)

**Lemma**

If $S$ is in solved form, then $\vec{S}$ is an idempotent mgu of $S$.

*Proof.* Suppose $\vec{S} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

- $\vec{S}$ is idempotent because none of the $x_i$ appears in any of the $t_j$.
- $\vec{S} \in \mathcal{U}(S)$ because $\vec{S}(x_i) = t_i = \vec{S}(t_i)$.
- $\vec{S}$ is an mgu because $\vec{S} \lesssim \sigma$ for any $\sigma \in \mathcal{U}(S)$.

# Transformation into Solved Form

## Transformation Rules

| | | | |
|---|---|---|---|
| DELETE | $\{t =^? t\} \uplus S$ | $\Longrightarrow$ | $S$ |
| DECOMPOSE | $\{f(\overline{t_n}) =^? f(\overline{u_n})\} \uplus S$ | $\Longrightarrow$ | $\{t_1 =^? u_1, \ldots, t_n =^? u_n\} \cup S$ |
| ORIENT | $\{t =^? x\} \uplus S$ | $\Longrightarrow$ | $\{x =^? t\} \cup S$ if $t \notin X$ |
| ELIMINATE | $\{x =^? t\} \uplus S$ | $\Longrightarrow$ | $\{x =^? t\} \cup \{x \mapsto t\}(S)$ if $x \in \mathcal{V}ar(S)$ and $x \notin \mathcal{V}ar(t)$ ("occurs check") |

- 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

$$\{x =^? f(a), g(x, x) =^? g(x, y)\} \quad \Longrightarrow \textsc{Eliminate}$$
$$\{x =^? f(a), g(f(a), f(a)) =^? g(f(a), y)\} \quad \Longrightarrow \textsc{Decompose}$$
$$\{x =^? f(a), f(a) =^? f(a), f(a) =^? y\} \quad \Longrightarrow \textsc{Delete}$$
$$\{x =^? f(a), f(a) =^? y\} \quad \Longrightarrow \textsc{Orient}$$
$$\{x =^? f(a), y =^? f(a)\}$$

# Example (2)

## Failure

$$\{f(x,x) =^? f(y,g(y))\} \Longrightarrow \textsc{Decompose}$$
$$\{x =^? y, x =^? g(y)\} \Longrightarrow \textsc{Eliminate}$$
$$\{x =^? y, y =^? g(y)\}$$

- No transformation rule is applicable to $\{x =^? y, y =^? g(y)\}$.
- ELIMINATE is not applicable to $y =^? g(y)$ because the occurs check fails.
- Dropping the occurs check can cause looping:
  $\{y =^? g(y), \ldots y \ldots\} \Longrightarrow$
  $\{y =^? g(y), \ldots g(y) \ldots\} \Longrightarrow$
  $\{y =^? g(y), \ldots g(g(y)) \ldots\} \Longrightarrow \ldots$

# Unification Algorithem

## Definition

$$\begin{array}{l}
\textit{Unify(S)} = \texttt{while} \text{ there is some } T \text{ such that } S \Longrightarrow T \texttt{ do} \\
\qquad\qquad S := T; \\
\qquad\quad \texttt{end while} \\
\qquad\quad \texttt{if } S \text{ is in solved form } \texttt{then} \text{ return } \vec{S} \texttt{ else} \text{ fail}
\end{array}$$

# Properties of Unify

- *Unify* is nondeterministic:
  If more than one transformation rule is applicable, say $S \Longrightarrow T_1$ and $S \Longrightarrow T_2$, then *Unify* may choose arbitrarily between $T_1$ and $T_2$.
- *Unify* is sound:
  If *Unify(S)* returns a substitution $\sigma$, then $\sigma$ is an idempotent mgu of $S$.
- *Unify* is complete:
  If a unification problem $S$ is solvable then *Unify(S)* does not fail.
- *Unify* terminates for all inputs.

# Proving Soundness (1)

UNI FREIBURG

**Lemma**

If $S \Longrightarrow T$ then $\mathcal{U}(S) = \mathcal{U}(T)$.

*Proof.* Case distinction on the rule used to transform $S$ to $T$:

- DELETE, DECOMPOSE, or ORIENT: obvious.
- ELIMINATE: $\{x =^? t\} \uplus S' \Longrightarrow \{x =^? t\} \cup \theta(S')$ with $\theta = \{x \mapsto t\}$ and $x \notin \mathcal{V}ar(t)$.
  - $\{x =^? t\}$ is in solved form, so $\sigma = \sigma\theta$ if $\sigma(x) = \sigma(t)$ by the lemma on page 13.
  - We now conclude:

$$
\begin{aligned}
\sigma \in \mathcal{U}(\{x =^? t\} \uplus S') &\Leftrightarrow \sigma(x) = \sigma(t) \text{ and } \sigma \in \mathcal{U}(S') \\
&\Leftrightarrow \sigma(x) = \sigma(t) \text{ and } \sigma\theta \in \mathcal{U}(S') \\
&\Leftrightarrow \sigma(x) = \sigma(t) \text{ and } \sigma \in \mathcal{U}(\theta S') \\
&\Leftrightarrow \sigma \in \mathcal{U}(\{x =^? t\} \cup \theta(S'))
\end{aligned}
$$

# Proving Soundness (2)

**Lemma**

If *Unify(S)* returns a substitution $\sigma$, then $\sigma$ is an idempotent mgu of $S$.

*Proof.*

- If *Unify(S)* returns a substitution $\sigma$, then *Unify* transforms $S$ until it reaches a unification problem $T$ in solved form. We then have $\sigma = \vec{T}$.
- Using the lemma just shown, we get $\mathcal{U}(S) = \mathcal{U}(T)$. (Proving this requires a straightforward induction on the number of transformation steps.)
- By using the lemma on page 14, we get that $\vec{T}$ is an idempotent mgu of $T$ and so it is also an idempotent mgu of $S$.

# Proving Completeness (1)

**Lemma**

An equation $f(s_1, \ldots, s_m) =^? g(t_1, \ldots, t_n)$ where $f \neq g$ has no solution.

**Lemma**

An equation $x =^? t$, where $x$ occurs in $t$ and $x \neq t$, has no solution.

*Proof.*

- If $x \neq t$ then $t = f(t_1, \ldots, t_n)$ and $x$ occurs in some $t_i$.
- Hence, $\sigma(x) \neq \sigma(t)$ for any substitution $\sigma$ because $|\sigma(x)| \leq |\sigma(t_i)| < |\sigma(t)|$.

# Proving Completeness (2)

UNI FREIBURG

**Lemma**

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

*Proof.* *Unify(S)* reduces $S$ to a normal form $T$ with respect to $\Longrightarrow$. By the lemma on page 20, $T$ is solvable. Also, $T$ cannot contain equations of the following form:

- $f(\dots) =^? f(\dots)$ (apply DECOMPOSE)
- $f(\dots) =^? g(\dots)$ ($T$ is solvable)
- $x =^? x$ (apply DELETE)
- $t =^? x$ where $t \notin X$ (apply ORIENT)

Hence, all equations of $T$ are of the form $x =^? t$. Additionally,

- $x \notin \mathcal{V}ar(t)$ ($T$ is solvable)
- $x$ cannot occur twice in $T$ (apply ELIMINATE)

This proves that $T$ is in solved form.

# General Strategy for Termination Proofs

To show that a reduction relation $(A, \to)$ terminates, we need to show that there are no infinite chains $x_1 \to x_2 \to x_3 \to \ldots$.

## Strategy

- Choose another reduction relation $(B, >)$ that is known to terminate.
- Associate every $x \in A$ with a measure $\varphi(x) \in B$.
- Prove that $x \to y$ implies $\varphi(x) > \varphi(y)$.

Now suppose there is an infinite chain in $A$

$$x_1 \to x_2 \to x_3 \to \ldots$$

This implies that there is an infinite chain in $B$

$$\varphi(x_1) > \varphi(x_2) > \varphi(x_3) > \ldots$$

But this is a contradiction because $>$ terminates!

# Lexicographic Orders (1)

**Definition**

A strict order on some set $A$ is a transitive and irreflexive relation on $A$.

**Definition**

Given two strict orders $(A, >_A)$ and $(B, >_B)$, the lexicographic product $>_{A \times B}$ on $A \times B$ is defined as follows:

$$(x, y) >_{A \times B} (x', y') \text{ iff } (x >_A x') \text{ or } (x = x' \text{ and } y >_B y')$$

# Lexicographic Orders (2)

**Lemma**

The lexicographic product of two strict orders is again a strict order.

*Proof.* Exercise.

**Lemma**

The lexicographic product of two terminating relations is again terminating.

*Proof.* By contradiction.

- Assume $(a_0, b_0) > (a_1, b_1) > (a_2, b_2) > \dots$
- This implies $a_0 \geq a_1 \geq a_2 \geq \dots$
- Because $>_A$ terminates there must be a $k \in \mathbb{N}$ such that $a_i = a_{i+1}$ for all $i \geq k$.
- This implies $b_k > b_{k+1} > b_{k+2} > \dots$ but $>_B$ terminates!

# Reduction Relations that Terminate

- The relation $>$ on $\mathbb{N}$ terminates.
- The relation $>_{\mathbb{N}\times\mathbb{N}}$ on $\mathbb{N} \times \mathbb{N}$ terminates.
- The relation $>_{\mathbb{N}\times\mathbb{N}\times\mathbb{N}}$ on $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ terminates.

# Proving Termination of Unify (1)

**Lemma**

*Unify* terminates for all inputs.

*Proof.* We call a variable $x$ solved in $S$ if $x$ occurs exactly once in $S$, namely on the left-hand side of some equation $x =^? t$.

We now prove termination of $\Longrightarrow$ by a measure function that maps a unification problem $S$ to a triple $(n_1, n_2, n_3)$ of natural numbers:

- $n_1$ is the number of variables in $S$ that are not solved.
- $n_2$ is the size of $S$, defined as $|S| := \Sigma_{(s=^?t)\in S}(|s| + |t|)$
- $n_3$ is the number of equations of the form $t =^? x$ in $S$.

It remains to be shown that each step of $\Longrightarrow$ decreases the triples with respect to the lexicographic ordering.

# Proving Termination of Unify (2)

| | $n_1$ #(unsolved variables) | $n_2$ size of $S$ | $n_3$ #(equations of the form $t =^? x$) |
|---|---|---|---|
| DELETE<br>$\{t =^? t\} \uplus S \Longrightarrow S$ | $\geq$ | $>$ | |
| DECOMPOSE<br>$\{f(\overline{t_n}) =^? f(\overline{u_n})\} \uplus S \Longrightarrow$<br>$\{t_1 =^? u_1, \ldots, t_n =^? u_n\} \cup S$ | $\geq$ | $>$ | |
| ORIENT<br>$\{t =^? x\} \uplus S \Longrightarrow$<br>$\{x =^? t\} \cup S$ if $t \notin X$ | $\geq$ | $=$ | $>$ |
| ELIMINATE<br>$\{x =^? t\} \uplus S \Longrightarrow$<br>$\{x =^? t\} \cup \{x \mapsto t\}(S)$<br>if $x \in \mathcal{V}ar(S)$<br>and $x \notin \mathcal{V}ar(t)$ | $>$ | | |

# Example

$$
\begin{array}{lll}
(2,9,0) & \{x =^? f(a), g(x,x) =^? g(x,y)\} & \Longrightarrow \text{ELIMINATE} \\
(1,12,0) & \{x =^? f(a), g(f(a),f(a)) =^? g(f(a),y)\} & \Longrightarrow \text{DECOMPOSE} \\
(1,10,1) & \{x =^? f(a), f(a) =^? f(a), f(a) =^? y\} & \Longrightarrow \text{DELETE} \\
(1,6,1) & \{x =^? f(a), f(a) =^? y\} & \Longrightarrow \text{ORIENT} \\
(0,6,0) & \{x =^? f(a), y =^? f(a)\} &
\end{array}
$$

# Earlier Failure Detection

- Detecting unsolvability can be expensive because *Unify* first computes a normal form.
- But if the unification problem contains
  - an equation $f(\dots) =^? g(\dots)$ with $f \neq g$ or
  - an equation $x =^? t$ with $x \in \mathcal{V}ar(t)$ 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:

$$
\begin{array}{lll}
\textsc{Clash} & \{f(\overline{t_n}) =^? g(\overline{u_n})\} \uplus S \Longrightarrow \perp & \text{if } f \neq g \\
\textsc{Occurs–Check} & \{x =^? t\} \uplus S \qquad\quad \Longrightarrow \perp & \\
& & \text{if } x \in \mathcal{V}ar(t) \\
& & \text{and } x \neq t
\end{array}
$$