# Foundations of Programming Languages and Software Engineering

Jan-Georg Smaus (Peter Thiemann)

Universität Freiburg

July 2011

- The Word Problem



## 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 $\sigma s \approx_{\mathcal{E}} \sigma t$.



## The Word Problem

**Definition**

Suppose $\Sigma$ is a signature and $X$ a set of variables disjoint from $\Sigma$.

- The (ground) word problem for $\mathcal{E}$ is the problem of deciding $s \approx_{\mathcal{E}} t$ for arbitrary $s, t \in T(\Sigma, \emptyset)$.



## Solving the Word Problem

**A Sample Problem**

Given $\Sigma_{int} = \{\mathtt{zero}^{(0)}, \mathtt{pred}^{(1)}, \mathtt{succ}^{(1)}\}$ and
$\mathcal{E}_{int} = \{\mathtt{pred}(\mathtt{succ}(x)) \approx x, \mathtt{succ}(\mathtt{pred}(x)) \approx x\}$
we would like to decide whether
$\mathtt{succ}(\mathtt{zero}) \approx_{\mathcal{E}_{int}} \mathtt{succ}(\mathtt{succ}(\mathtt{pred}(\mathtt{zero})))$

**A solution**

- Use identities as reduction rules:
  $\mathtt{pred}(\mathtt{succ}(x)) \rightarrow_{\mathcal{E}_{int}} x, \mathtt{succ}(\mathtt{pred}(x)) \rightarrow_{\mathcal{E}_{int}} x$
- Apply reduction rules to both terms:
  - $\mathtt{succ}(\underline{\mathtt{succ}(\mathtt{pred}(\mathtt{zero}))}) \rightarrow_{\mathcal{E}_{int}} \mathtt{succ}(\mathtt{zero})$
- Check whether the resulting terms are identical.

Problem: Applying the reduction rules might not terminate.

## Undecidability

The computation of any Turing machine can be simulated by an appropriate signature $\Sigma$ and set of identities $\mathcal{E}$ $\Rightarrow$ the word problem in general is undecidable.



## The Reduction Relation Generated by $\Sigma$-Identities

**Definition**

Let $\mathcal{E}$ be a set of $\Sigma$-identities.
The reduction relation $\rightarrow_{\mathcal{E}} \subseteq T(\Sigma, X) \times T(\Sigma, X)$ is defined as

$$s \rightarrow_{\mathcal{E}} t \text{ iff}$$

there exists $(l, r) \in \mathcal{E}$, $p \in Pos(s)$, and a substitution $\sigma$ with $s|_p = \sigma(l)$ and $t = s[\sigma(r)]_p$.



## Example

**Computing with Groups**

$$\Sigma_G = \{e^{(0)}, i^{(1)}, f^{(2)}\}$$

$$\begin{aligned}\mathcal{E}_G = \{ & f(x, f(y, z)) \approx f(f(x, y), z), \\ & f(e, x) \approx x, \\ & f(i(x), x) \approx e\}\end{aligned}$$

$$\begin{array}{lll} & f(i(e), f(e, e)) & \sigma_1 = \{x \mapsto i(e), y \mapsto e, z \mapsto e\}, 1^{st} \text{ id} \\ \rightarrow_{\mathcal{E}_G} & f(f(i(e), e), e) & \sigma_2 = \{x \mapsto e\}, 3^{rd} \text{ id} \\ \rightarrow_{\mathcal{E}_G} & f(e, e) & \sigma_3 = \{x \mapsto e\}, 2^{nd} \text{ id} \\ \rightarrow_{\mathcal{E}_G} & e & \end{array}$$



## Composing Relations

**Definition**

Given two relations $R \subseteq A \times B$ and $S \subseteq B \times C$, their composition is defined by

$$S \circ R := \{(x, z) \in A \times C \mid \text{there exists some } y \in B \text{ with } (x, y) \in R \text{ and } (y, z) \in S\}$$

**Example**

Suppose $R = \{\text{FR} \rightarrow \text{OG}, \text{OG} \rightarrow \text{KA}, \text{KA} \rightarrow \text{MA}\}$.
Then $R \circ R = \{\text{FR} \rightarrow \text{KA}, \text{OG} \rightarrow \text{MA}\}$.

# Notations for Reduction Relations

Suppose $\rightarrow$ is a binary relation on $M$.

| | |
|---|---|
| $\xrightarrow{0} := \{(x,x) \mid x \in M\}$ | identity |
| $\xrightarrow{i+1} := \rightarrow \circ \xrightarrow{i}$ | $(i+1)$-fold composition, $i \geq 0$ |
| $\xrightarrow{+} := \bigcup_{i>0} \xrightarrow{i}$ | transitive closure |
| $\xrightarrow{*} := \xrightarrow{+} \cup \xrightarrow{0}$ | reflexive transitive closure |
| $\leftarrow := \{(y,x) \mid x \rightarrow y\}$ | inverse |



# Terminology for Reduction Relations (1)

Suppose $\rightarrow$ is a binary relation on $M$ and $x, y \in M$.

- $x$ is reducible iff there is a $z \in M$ with $x \rightarrow z$.
- $x$ is in normal form iff it is not reducible.
- $y$ is a normal form of $x$ iff $x \xrightarrow{*} y$ and $y$ is in normal form.
- if $x$ has a unique normal form, it is denoted by $x{\downarrow}$.
- $x$ and $y$ are joinable iff there is a $z \in M$ such that $x \xrightarrow{*} z \xleftarrow{*} y$. We then write $x \downarrow y$.

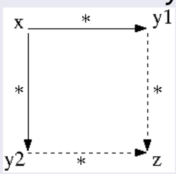


# Terminology for Reduction Relations (2)

**Definition**

A reduction $\rightarrow$ is called

- confluent iff $y_1 \xleftarrow{*} x \xrightarrow{*} y_2$ implies $y_1 \downarrow y_2$,




- terminating iff there is no infinite chain $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \dots$.



# Deciding the Word Problem

**Theorem (deciding the word problem for $\mathcal{E}$)**

If $\mathcal{E}$ is finite and $\rightarrow_{\mathcal{E}}$ is confluent and terminating, then the word problem for $\mathcal{E}$ is decidable.

- Plan: To decide whether $s \approx_{\mathcal{E}} t$ holds, compare $s{\downarrow_{\mathcal{E}}}$ and $t{\downarrow_{\mathcal{E}}}$ for syntactic equality.
- Caveat:
  - $s{\downarrow_{\mathcal{E}}}$ and $t{\downarrow_{\mathcal{E}}}$ must exist
  - $s{\downarrow_{\mathcal{E}}}$ and $t{\downarrow_{\mathcal{E}}}$ must be computable
- We do not give the proof details here, but some important facts are ...

## Existence and Uniqueness of Normal Forms

- If $\to$ is confluent, every element has at most one normal form.
- If $\to$ is terminating, every element has at least one normal form.

$\Rightarrow$ If $\to$ is confluent and terminating, every element has a unique normal form.



## Deciding the Word Problem

**Theorem (deciding the word problem for $\mathcal{E}$)**

If $\mathcal{E}$ is finite and $\to_{\mathcal{E}}$ is confluent and terminating, then the word problem for $\mathcal{E}$ is decidable.

*Proof.* Suppose $s, t \in T(\Sigma, X)$. We must give an algorithm that decides $s \approx_{\mathcal{E}} t$. Since $s \approx_{\mathcal{E}} t$ and $s{\downarrow_{\mathcal{E}}} = t{\downarrow_{\mathcal{E}}}$ are equivalent (proof omitted), we only need to give an algorithm for computing the normal form $u{\downarrow_{\mathcal{E}}}$ for any term $u$.



## Computing Normal Forms (1)

Suppose $\mathcal{E}$ is finite and $\to_{\mathcal{E}}$ is confluent and terminating. Given a term $u \in T(\Sigma, X)$, we can compute the normal form $u{\downarrow_{\mathcal{E}}}$ using the following iteration:

1. Decide if $u$ is already in normal form w.r.t $\to_{\mathcal{E}}$. If yes, stop. Otherwise, continue with step (2).
2. Find some $u'$ such that $u \to_{\mathcal{E}} u'$ (if $u$ is not in normal form). Then continue with step (1), setting $u = u'$.

This iteration terminates because $\to_{\mathcal{E}}$ is terminating.



## Computing Normal Forms (2)

Here is how we decide whether $u$ is in normal form:

- For all identities $(l, r) \in \mathcal{E}$ (only finitely many), and
- all positions $p \in Pos(u)$ (only finitely many)
- check whether there exists a substitution $\sigma$ such that $u|_p = \sigma(l)$. If yes, then we can reduce $u$ to $u[\sigma(r)]_p$. If not, $u$ is already in normal form.

We will see later that finding a substitution $\sigma$ such that $u|_p = \sigma(l)$ is also decidable.