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 $s, t \in T(\Sigma, \emptyset)$.

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

A solution
- Use identities as reduction rules:
  \[
  \text{pred}(\text{succ}(x)) \rightarrow_{\mathcal{E}_{int}} x, \text{succ}(\text{pred}(x)) \rightarrow_{\mathcal{E}_{int}} x
  \]
- Apply reduction rules to both terms:
  - $\text{succ}(\text{succ}(\text{pred}(\text{zero}))) \rightarrow_{\mathcal{E}_{int}} \text{succ}(\text{zero})$
- 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 $E \Rightarrow$ the word problem in general is undecidable.

Example

Computing with Groups

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

$E_A = \{ f(x, f(y, z)) \approx f(f(x, y), z), f(e, x) \approx x, f(i(x), x) \approx e \}$

$\rightarrow_{E_A} f(i(e), f(e, e)) \quad \sigma_1 = \{ x \mapsto i(e), y \mapsto e, z \mapsto e \}, 1^{st} \text{ id}$

$\rightarrow_{E_A} f(f(i(e), e), e) \quad \sigma_2 = \{ x \mapsto e \}, 3^{rd} \text{ id}$

$\rightarrow_{E_A} f(e, e) \quad \sigma_3 = \{ x \mapsto e \}, 2^{nd} \text{ id}$

$\rightarrow_{E_A} e$

The Reduction Relation Generated by $\Sigma$-Identities

Definition

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

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

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 $M$.

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

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 $x \rightarrow^* y$ 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 $x \rightarrow^* z \leftarrow y$. We then write $x \downarrow y$.

Terminology for Reduction Relations (2)

Definition
A reduction $\rightarrow$ is called
- confluent iff $y_1 \leftarrow x \rightarrow y_2$ implies $y_1 \downarrow y_2$.
- terminating iff there is no infinite chain $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots$.

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 → is confluent, every element has at most one normal form.
- If → is terminating, every element has at least one normal form.
⇒ If → is confluent and terminating, every element has a unique normal form.

Deciding the Word Problem

Theorem (deciding the word problem for \( E \))

If \( E \) is finite and \( \rightarrow_E \) is confluent and terminating, then the word problem for \( E \) is decidable.

Proof. Suppose \( s, t \in T(\Sigma, X) \). We must give an algorithm that decides \( s \approx_E t \). Since \( s \approx_E t \) and \( s \downarrow_E = t \downarrow_E \) are equivalent (proof omitted), we only need to give an algorithm for computing the normal form \( u \downarrow_E \) for any term \( u \).

Computing Normal Forms (1)

Suppose \( E \) is finite and \( \rightarrow_E \) is confluent and terminating. Given a term \( u \in T(\Sigma, X) \), we can compute the normal form \( u \downarrow_E \) using the following iteration:

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

This iteration terminates because \( \rightarrow_E \) is terminating.

Computing Normal Forms (2)

Here is how we decide whether \( u \) is in normal form:

- For all identities \( (l, r) \in E \) (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.