For ground identities, the word problem is decidable by congruence closure. Now we want to regard the general case where $e, s, t$ can contain variables.

By Birkhoff’s Theorem: To check $s \equiv e \cdot t$, one has to investigate $s \xrightarrow{e} e \cdot t$. This has an enormous search space.

To reduce the search space: restrict orientation of equations such that they can only be applied from left to right.

**Def 3.3.1. (Term Rewriting System, TRS)**

For $l, r \in \tau(\Sigma, \mathcal{V})$, we say that $l \rightarrow r$ is a **rule** over $\Sigma$ and $\mathcal{V}$ iff

- $\mathcal{V}(r) \subseteq \mathcal{V}(l)$
- $l \neq \tau$

A set $\mathcal{R}$ of rules is called a **term rewriting system (TRS)**. For a TRS $\mathcal{R}$, the rewrite relation $\rightarrow^*_{\mathcal{R}} \subseteq \tau(\Sigma, \mathcal{V}) \times \tau(\Sigma, \mathcal{V})$ is defined as:

$$s \rightarrow^*_{\mathcal{R}} t \quad \text{iff} \quad s \mid_\pi = l \circ \tau \quad \text{and} \quad t = s \tau \circ \tau_\pi$$

for some $\tau \in \text{O}e(s)$, some $l \rightarrow \tau \in \mathcal{R}$, some $\tau \in \text{SUB}(\Sigma, \mathcal{V})$.

The subterm $s \mid_\pi$ is called a **redex** ("reducible expression").

Sometimes we write $s \rightarrow t$ instead of $s \rightarrow^*_{\mathcal{R}} t$ if $\mathcal{R}$ is clear from the context.
Reason for "\( W(Y) \leq W(K) \)" : this restricts the search space. Now the matcher of \( K \) uniquely determines the instantiation of \( X \). For any term \( s \), there are only finitely many terms \( t \) with \( s \rightarrow_{\Sigma} t \) (if \( \Sigma \) is finite). \( \Rightarrow \) Breadth of search tree for the word problem is finite.

Reason for "\( L \neq W \)" : also restricts the search space.

A rule \( x \rightarrow t \) would always be applicable, since a variable \( x \) matches any term.

**Ex. 3.32.** We now want to use \( T(R, \Sigma) \) instead of sets of equations \( \Sigma \).

For groups:
\[
\begin{align*}
f(x, f(y, z)) & \rightarrow f(f(x, y), z) \\
f(x, e) & \rightarrow x \\
f(x, i(x)) & \rightarrow e
\end{align*}
\]

Similarly for addition:
\[
\begin{align*}
\text{plus}(s(s(0)), s(0)) & \rightarrow s(\text{plus}(s(0), s(0))) \\
& \rightarrow s(s(s(0))) \\
& \rightarrow s(s(s(0)))
\end{align*}
\]

Term Rewriting is very simple, but it is already a
Turing-complete programming language (i.e., every computable function can be computed by a TRS).

Clearly: \(\rightarrow^*\) is stable and monotonic

\(\rightarrow\) is reflexive and transitive (\(+\) stable, \(+\) monotonic)

\(\leftarrow\) is symmetric and \(\rightarrow\)

To solve the word problem for \(\mathcal{E}\):

* transform set of equations \(\mathcal{E}\) into an equivalent TRS \(\mathcal{R}\)

* use \(\mathcal{R}\) to solve the word problem

**Definition 3.3.3 (Equivalence of Set of Equations and TRS)**

Let \(\mathcal{E}\) be a set of equations and \(\mathcal{R}\) be a TRS. Then \(\mathcal{R}\) is equivalent to \(\mathcal{E}\) iff \(\mathcal{R} \leftrightarrow^* \mathcal{E}\).

Reason: \(\mathcal{S} \mathcal{E}\) iff \(\mathcal{S} \mathcal{R}\) iff \(\mathcal{S} \mathcal{E}\). Birkhoff’s Theorem

If \(\mathcal{R}\) and \(\mathcal{E}\) are equivalent

To check whether \(\mathcal{R}\) is equivalent to \(\mathcal{E}\), it suffices
to regard $\equiv$ and $\rightarrow_{\equiv}$ instead of $\rightarrow_{\text{E}}$ and $\rightarrow_{\text{R}}$.

**Thm 334** (Connection between Sets of Equations and TRSs)

Let $\equiv$ be a set of equations and $\rightarrow_{\text{R}}$ be a TRS. $\equiv$ is equivalent to $\rightarrow_{\text{E}}$ iff

1. $l \rightarrow_{\equiv} r$ for all rules $l \rightarrow_r r \in \rightarrow_{\text{E}}$ ("$\rightarrow_{\text{R}}$ is sound for $\equiv$")
2. $S \rightarrow_{\equiv} t$ for all equations $S \equiv t \in \equiv$ ("$\equiv$ is adequate for $\equiv$")

**Proof:**

$\Rightarrow$: Equivalence trivially implies that $\equiv$ is sound and adequate for $\rightarrow_{\text{E}}$.

$\Leftarrow$: Equivalence follows from soundness and adequateness: Follows directly from the fact that $\rightarrow_{\text{E}}$ and $\rightarrow_{\text{R}}$ are stable, monotonic, reflexive, transitive, and symmetric.

If one simply replaces "$\equiv$" by "$\rightarrow$" in $\equiv$, then one clearly obtains an equivalent TRS. But these TRSs are not always advantageous, i.e., we sometimes want other TRSs that are still equivalent to $\equiv$.
\( \Sigma = \{ b = c, b = a, f(a) = f(f(a)) \} \)

\( \Sigma = \{ a, b, c, f \} \)

\( \mathcal{R} = \{ c \rightarrow b, a \rightarrow b, f(f(b)) \rightarrow f(b) \} \)

\( \mathcal{R} \) is sound for \( \Sigma \), because:

- \( c \not\in \mathcal{R} \)
- \( a \not\in \mathcal{R} \)
- \( \mathcal{R} (f(b)) \rightarrow f(f(a)) \not\in \mathcal{R} \)
- \( f(a) \in \mathcal{R} \)
- \( f(b) \in \mathcal{R} \)

\( \mathcal{R} \) is adequate for \( \Sigma \), because:

- \( b \not\in \mathcal{R} \)
- \( a \not\in \mathcal{R} \)
- \( \mathcal{R} (f(b)) \rightarrow f(f(a)) \)
- \( \mathcal{R} (f(a)) \not\in \mathcal{R} \)
- \( \mathcal{R} (f(f(b))) \not\in \mathcal{R} \)

Thus: \( \mathcal{R} \) is equivalent to \( \Sigma \).

Why don’t we just take the equations \( \Sigma \) and replace “=” by “\( \rightarrow \)” in order to obtain an equivalent \( \mathcal{R} \)?

Reason: The rules of \( \mathcal{R} \) should not be applied in both directions (in order to reduce the search space).

Goal: To solve \( s = \Sigma t \), choose an equivalent
"Suitable" TRS $\mathcal{R}$.

To check $S \xrightarrow{\mathcal{R}}^* T$, we want to proceed as follows:

- Rewrite both $S$ and $T$ as long as possible:

  \[ S \xrightarrow{\mathcal{R}} S_1 \xrightarrow{\mathcal{R}} S_2 \xrightarrow{\mathcal{R}} \ldots \xrightarrow{\mathcal{R}} S_n \]
  \[ T \xrightarrow{\mathcal{R}} T_1 \xrightarrow{\mathcal{R}} T_2 \xrightarrow{\mathcal{R}} \ldots \xrightarrow{\mathcal{R}} T_m \]

- Check whether the resulting terms $S_n$ and $T_m$ are syntactically equal.
  - If yes: return "true" (i.e., $S \equiv T$ holds)
  - If no: return "false" (i.e., $S \equiv T$ does not hold).

**Ex 336** Use the plus-TRS to check

\[ \text{plus}(s(s(\varnothing)), x) \equiv_\mathcal{R} \text{plus}(s(\varnothing), s(x)) \]

\[ \text{plus}(s(s(\varnothing)), x) \xrightarrow{\mathcal{R}} s(\text{plus}(s(\varnothing), x)) \xrightarrow{\mathcal{R}} s(s(\text{plus}(\varnothing, x))) \xrightarrow{\mathcal{R}} s(s(x)) \]

\[ \text{plus}(s(\varnothing), s(x)) \xrightarrow{\mathcal{R}} s(\text{plus}(\varnothing, s(x))) \xrightarrow{\mathcal{R}} s(s(x)) \]

Alg. returns "true".

If the algorithm returns "true", then we really have
$S \equiv \varepsilon$. But the algorithm has 2 problems:

1. It can happen that the reduction of $S$ or $T$ does not terminate (this can also happen if $S \equiv \varepsilon$, i.e., the algorithm is not even a semi-decision procedure for the word problem).

   Solution: require that $S$ is terminating.

2. It can happen that $S \not\Rightarrow^* T$ holds, but there is no term $\eta$ such that $S \Rightarrow^* \eta \not\Rightarrow^* T$.
   So the alg. could return "False" although $S \equiv \varepsilon T$ holds.

   Solution: require that $S$ is confluent.

Ex 337

$E = \{ b \equiv c, \ b \equiv a, \ f(a) = f(f(a)) \}$

$\mathcal{R}_a = \{ b \rightarrow c, \ b \rightarrow a, \ f(a) \rightarrow f(f(a)) \}$ is clearly equivalent to $E$.

Check whether $f(a) \equiv \varepsilon f(c)$ holds.

   $f(a) \rightarrow \ f(f(a)) \rightarrow \ f(f(f(a))) \rightarrow \ f^{4}(a) \rightarrow \ \ldots$

   $\Rightarrow$ Alg. does not terminate.

   Indeed, $\mathcal{R}_a$ is not terminating.

$\mathcal{R}_2 = \{ b \rightarrow c, \ b \rightarrow a, \ f(f(a)) \rightarrow f(a) \}$ is equivalent to $E$.
Check whether \( f(a) \equiv_\varepsilon f(c) \).

Neither \( f(a) \) nor \( f(c) \) can be reduced further.

\( \Rightarrow \) Alg. returns "False" since \( f(a) \) and \( f(c) \) are not syntactically equal. (This is the wrong answer, since \( f(a) \equiv_\varepsilon f(c) \) holds.)

Indeed, \( R_2 \) is not confluent.