Equational logic, unification
and term rewriting

1 Equational logic

Below we shall largely follow Klop (1992) in the presentation of equational logic
and unification.

1.1 Some notions from universal algebra

In universal algebra properties of general algebraic systems are studied. These
systems include the usual, groups, semigroups, monoids, rings, but also sys-
tems with operations of arbitrary number of arguments. In algebraic specification
theory these operations may describe programs or hardware components. (See

A signature $\Sigma$ is a set of function symbols, where each $F \in \Sigma$ takes a fixed
number $n(F)$ (the arity) of arguments. 0-ary function symbols are considered as
constant symbols. (Thus a signature is like a description of a first order language
but without relation symbols.) A $\Sigma$-algebra $A$ consists of an underlying nonempty
set $A$, and for each function symbol $F \in \Sigma$, an operation

$$F^A : A^{n(F)} \to A,$$

for $n(F) > 0$. If $n(F) = 0$, $F^A \in A$.

Homomorphisms, mappings which preserves the operations of an algebra are
of central importance. Let $A$ and $B$ be $\Sigma$-algebras. A ($\Sigma$-algebra) homomorphism
$\phi : A \to B$ is function between the underlying sets $\phi : A \to B$ which is such that
for every function symbol $F \in \Sigma$ of arity $n$ we have for all $a_1, \ldots, a_n \in A$:

$$\phi(F^A(a_1, \ldots, a_n)) = F^B(\phi(a_1), \ldots, \phi(a_n)).$$

If $n = 0$, this reads $\phi(F^A) = F^B$. 
Example 1.1 The embedding $\mathbb{Z} \hookrightarrow \mathbb{Q}$ and the quotient mapping

$$x \mapsto x \mod n : \mathbb{Z} \to \mathbb{Z}/n\mathbb{Z}$$

are basic examples of homomorphisms with respect to the signature $\Sigma = \{0, 1, +, \cdot\}$.

There is always a trivial homomorphism $\mathcal{A} \to \mathcal{A}$, the identity homomorphism $\text{id}_\mathcal{A}$ defined by $\text{id}_\mathcal{A}(x) = x$. A homomorphism $\varphi : \mathcal{A} \to \mathcal{B}$ is an isomorphism if there is a homomorphism $\psi : \mathcal{B} \to \mathcal{A}$ such that $\psi \circ \varphi = \text{id}_\mathcal{A}$ and $\varphi \circ \psi = \text{id}_\mathcal{B}$. We leave the verification of the following result to the reader:

Proposition 1.2 A homomorphism $\varphi : \mathcal{A} \to \mathcal{B}$ is an isomorphism iff $\varphi : \mathcal{A} \to \mathcal{B}$ is a bijection. $\square$

1.2 Terms and termalgebras

Let $\text{Ter}(\Sigma)$ be the set of terms that can be formed from the function symbols in $\Sigma$ and variables from a fixed set of variable symbols $\mathcal{X} = \{x_1, x_2, x_3, \ldots\}$. The set $\text{Ter}(\Sigma)$ is inductively defined by the following clauses

(T1) If $x \in \mathcal{X}$, then $x \in \text{Ter}(\Sigma)$.

(T2) If $F \in \Sigma$ and $n(F) = 0$, then $F \in \text{Ter}(\Sigma)$.

(T3) If $F \in \Sigma$, $n = n(F) > 0$ and $t_1, \ldots, t_n \in \text{Ter}(\Sigma)$, then $F(t_1, \ldots, t_n) \in \text{Ter}(\Sigma)$.

Since the set $\text{Ter}(\Sigma)$ is inductively defined, we may prove properties of terms by structural induction. We may also define functions on terms by structural recursion. A substitution is a function $\sigma : \mathcal{X} \to \text{Ter}(\Sigma)$, assigning to each variable symbol a term. The effect $t^\sigma$ of a substitution $\sigma$ on a term $t$ is defined recursively

$$
\begin{align*}
x_i^\sigma &= \sigma(x_i) \\
F^\sigma &= F \quad (n(F) = 0) \\
F(t_1, \ldots, t_n)^\sigma &= F(t_1^\sigma, \ldots, t_n^\sigma) \quad (n = n(F))
\end{align*}
$$

Thus we may extend $\sigma$ to a function $\text{Ter}(\Sigma) \to \text{Ter}(\Sigma)$ by $\sigma(t) = t^\sigma$. For finite substitutions we introduce a special notation. Denote by

$$\{x_{i_1} := t_1, \ldots, x_{i_k} := t_k\},$$

where $i_1 < i_2 < \cdots < i_k$, the substitution $\sigma$ where $\sigma(x_{i_j}) = t_j$ for $j = 1, \ldots, k$ and $\sigma(x_i) = x_i$ for $i \notin \{i_1, i_2, \ldots, i_k\}$. 

2
Example 1.3 Let $\Sigma = \{0, f, g\}$ where the arities are $n(0) = 0$, $n(f) = 1$ and $n(g) = 2$. Then $0$, $f(0), g(x_1, f(x_3))$ are examples of terms over $\Sigma$. For the substitution $\sigma = \{x_1 := g(x_1, x_3), x_2 := f(0), x_3 := x_2\}$ we have

$$g(x_1, f(x_3))^{\sigma} = g(x_1^{\sigma}, f(x_3^{\sigma})) = g(g(x_1, x_3), f(x_2)). \square$$

The set $\text{Ter}(\Sigma)$ can be regarded as a $\Sigma$-algebra — in a kind of trivial way — by defining for each $n$-ary function symbol $F \in \Sigma$, a function $F^{\text{Ter}(\Sigma)}$ by

$$F^{\text{Ter}(\Sigma)}(t_1, \ldots, t_n) = F(t_1, \ldots, t_n).$$

We call $\text{Ter}(\Sigma)$ the term algebra of $\Sigma$. We may also restrict ourselves to terms without variables (in case there are constant symbols) The resulting set, $\text{Ter}_0(\Sigma)$, also forms a $\Sigma$-algebra.

Note that any substitution $\sigma : \mathbb{X} \to \text{Ter}(\Sigma)$ extends to a $\Sigma$-algebra homomorphism $\sigma : \text{Ter}(\Sigma) \to \text{Ter}(\Sigma)$. (Exercise: verify this.)

Let $\mathcal{A}$ be a $\Sigma$-algebra. A variable assignment or environment in $\mathcal{A}$ is a function $\rho: \mathbb{X} \to \mathcal{A}$. Given such an assignment, the value $[[t]]^A_\rho$ of a term $t$ in $\mathcal{A}$ is determined. Define by recursion on $t$:

$$[[x_i]]^\rho = \rho(x_i),$$
$$[[F(t_1, \ldots, t_m)]]^\rho = F^\mathcal{A}([[t_1]]^\rho, \ldots, [[t_m]]^\rho).$$

An equation $s = t$ is valid in $\mathcal{A}$ (in symbols: $\mathcal{A} \models s = t$) iff for all variable assignments $\rho$ in $\mathcal{A}$: $[[s]]^\mathcal{A}_\rho = [[t]]^\mathcal{A}_\rho$.

An equational theory over $\Sigma$ is given by a set $E$ of equations $s = t$ where $s, t \in \text{Ter}(\Sigma)$. The deduction rules of an equational theory essentially only tell how instances of these equations may be used to calculate inside terms. We denote by $E \vdash_{\text{eq}} s = t$ that $s = t$ is derivable from $E$. The deduction rules are more formally

$$\frac{E \vdash_{\text{eq}} s = t}{E \vdash_{\text{eq}} s^{\sigma} = t^{\sigma}} \quad \text{(subst)} \quad \text{for every substitution } \sigma : \mathbb{X} \to \text{Ter}(\Sigma)$$

$$\frac{E \vdash_{\text{eq}} s_1 = t_1 \cdots E \vdash_{\text{eq}} s_n = t_n}{E \vdash_{\text{eq}} F(s_1, \ldots, s_n) = F(t_1, \ldots, t_n)} \quad \text{(cong)} \quad \text{for every } F \in \Sigma \text{ with } n = n(F)$$

$$\frac{E \vdash_{\text{eq}} t = t}{E \vdash_{\text{eq}} s^{\sigma} = t^{\sigma}} \quad \text{(refl :)}$$

$$\frac{E \vdash_{\text{eq}} s = t}{E \vdash_{\text{eq}} t = s} \quad \text{(symm)}$$

$$\frac{E \vdash_{\text{eq}} s = v \quad E \vdash_{\text{eq}} v = t}{E \vdash_{\text{eq}} s = t} \quad \text{(trans)}$$

3
**Example 1.4** *The equational theory of groups.* Let $\Sigma = \{1, \cdot, (\_)^{-1}\}$, where the arities are 0, 2 and 1 respectively. The equations $E$ are

$$1 \cdot x_1 = x_1, \quad x_1 \cdot 1 = x_1,$$

$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3,$$

$$x_1 \cdot x_1^{-1} = 1, \quad x_1^{-1} \cdot x_1 = 1.$$ 

We give an example of a formal derivation of $x_2 \cdot 1^{-1} = x_2$ from the axioms of $E$:

\[
\begin{align*}
\frac{x_1 \cdot 1 = x_1}{1^{-1} \cdot 1 = 1^{-1} \cdot 1} & \quad \text{(subst)} \\
\frac{1^{-1} \cdot 1 = 1^{-1} \cdot 1}{1^{-1} = 1^{-1} \cdot 1} & \quad \text{(symm)} \\
\frac{1^{-1} \cdot x_1 = 1}{1^{-1} \cdot 1 = 1} & \quad \text{(subst)} \\
\frac{x_2 = x_2}{x_2 \cdot 1^{-1} = x_2 \cdot 1} & \quad \text{(cong)} \\
\frac{x_2 \cdot 1^{-1} = x_2 \cdot 1}{x_2 \cdot 1^{-1} = x_2} & \quad \text{(trans)} \\
\frac{x_2 \cdot 1^{-1} = x_2}{x_2 \cdot 1 = x_2} & \quad \text{(subst)} \\
\frac{x_2 \cdot 1 = x_2}{x_2 \cdot 1 = x_2} & \quad \text{(trans)}
\end{align*}
\]

\[\square\]

A $\Sigma$-algebra $A$ is a model of $E$ (in symbols: $A \models E$) iff $A \models s = t$, for each $s = t \in E$. We say that $s = t$ is a (semantic) equational consequence of $E$ (in symbols: $E \models s =_{eq} t$) if for every $\Sigma$-algebra $A$:

$$A \models E \implies A \models s = t.$$ 

We now prove Birkhoff’s completeness theorem for equational theories. Let $=_{E}$ be the relation on $\text{Ter}(\Sigma)$ defined by

$$s =_{E} t \iff \text{def } E \vdash_{eq} s = t.$$ 

This relation of $E$-provable equality is an equivalence relation and a congruence with respect to the operations $F^{\text{Ter}(\Sigma)}$, according to the rules of the equational theory. We consider the set $T(E) = \text{Ter}(\Sigma)/=_{E}$ of equivalence classes $[t]$ of terms. Thus the following is a well-defined operation

$$F^{T(E)}([t_1], \ldots, [t_n]) = [F(t_1, \ldots, t_n)]$$ 

for any $F \in \Sigma$. Thus $T(E)$ is a $\Sigma$-algebra.

**Theorem 1.5 (Birkhoff)** Let $\Sigma$ be a signature and let $E$ be an equational theory over $\Sigma$. Then

$$E \vdash_{eq} s = t \iff T(E) \models s = t.$$ 

4
where for each \( t \) the \( (s, t) \) are arbitrary. The equational theory of Abelian groups.

Example 1.8 The equational theory of Abelian groups. Let \( \Sigma = \{ 1, \cdot, (\cdot)^{-1} \} \), where the arities are 0, 2 and 1 respectively. The equations E are

\[
\begin{align*}
1 \cdot x_1 &= x_1, \\
x_1 \cdot (x_2 \cdot x_3) &= (x_1 \cdot x_2) \cdot x_3, \\
x_1 \cdot x_2 &= x_2 \cdot x_1, \\
x_1 \cdot x_1^{-1} &= 1, \\
x_1^{-1} \cdot x_1 &= 1.
\end{align*}
\]

The models of this theory are exactly the Abelian groups. Denote by \( u^0 = 1 \) and \( u^{n+1} = u \cdot u^n \) for \( n \in \mathbb{N} \). For \( n > 0 \), let \( u^{-n} = (u^{-1})^n \). It is easy to show that for each \( t \in \text{Ter}(\Sigma) \) there are sequences \( n_1, \ldots, n_k \in \mathbb{Z} - \{0\} \), \( 1 \leq i_1 < i_2 \cdots < i_k \), where \( k \geq 0 \), such that

\[
t =_{E} x_{i_1}^{n_1} \cdot x_{i_2}^{n_2} \cdots \cdot x_{i_k}^{n_k}.
\] (1)
(In case \(k = 0\), the product is simply 1.) Thus in the model \(T(E)\) the equivalence classes are represented by elements of the form \(x_{i_1}^{n_1} \cdot x_{i_2}^{n_2} \cdots x_{i_k}^{n_k}\). \(\Box\)

One can in fact show that the sequences \((n_j, i_j)\) in (1) are unique. This can be used to decide when two terms are provably equal. A systematic method for obtaining such decidability results is provided by the theory of term rewriting systems.

For a signature \(\Sigma\) with at least one constant symbol, consider \(T_0(E)\) which is defined as \(T(E)\) but \(\text{Ter}_0(\Sigma)\) is used instead of \(\text{Ter}(\Sigma)\). (Exercise: What is \(T_0(E)\) in the case of Example 1.8? If new constants are added?)

**Theorem 1.9** Let \(E\) be an equational theory over a signature \(\Sigma\), which has at least one constant symbol. Then

1. \(T_0(E) \models E\)
2. if \(A \models E\), there is a unique homomorphism \(\varphi : T_0(E) \rightarrow A\).

**Proof.** (a): This is proved as in the direction \((\Rightarrow)\) of Theorem 1.5, but using \(T_0(E)\) instead of \(T(E)\).

(b): Define \(\varphi : T_0(E) \rightarrow A\) by \(\varphi([t]) = [[t]]_\tau^A\) where \(\tau\) is some fixed variable assignment (it does not matter which since \(t\) has no variables). It is well-defined because if \([s] = [t]\), then \(E \vdash_{\text{eq}} s = t\). Now \(A \models E\), so \(A \models s = t\), and hence in particular \([[s]]_\tau^A = [[t]]_\tau^A\). Furthermore \(\varphi\) is a homomorphism, since

\[
\varphi(F^{T(E)}([t_1], \ldots, [t_n])) = \varphi(F(t_1, \ldots, t_n)) = \varphi(F([t_1], \ldots, [t_n])) = F^A([[t_1]]_\tau^A, \ldots, [[t_n]]_\tau^A) = F^A(\varphi([t_1]), \ldots, \varphi([t_n])).
\]

Now, if \(\psi\) were another homomorphism, it is easily shown that \(\psi([t]) = \varphi([t])\) by induction on \(t\). \(\Box\)

Because of this theorem the model \(T_0(E)\) is called the initial model of the theory \(E\).

**Remark 1.10** For algebraic specification of programs one usually consider \(\Sigma\)-algebras with many sorts (types). For instance, we may have a sort \(A\) for an alphabet and a sort \(S\) for a stack. The constants are \(a, b, c : A\) (the letters of the alphabet), \(\text{nil} : S\) (the empty stack), the function symbols are \(\text{pop} : S \rightarrow S\) and \(\text{push} : A \times S \rightarrow S\). The equations \(E\) are

\[
\text{pop}(\text{nil}) = \text{nil},
\]

\[
\text{pop}(\text{push}(x^A, t^S)) = t^S
\]
(Here $x^A, t^S$ indicate variables of the different sorts.) The definitions and results above easily extend to many-sorted $\Sigma$-algebras.

**Exercises**

1. Let $A^*$ be set of strings over the alphabet $A$. Describe this set as an algebra with a binary concatenation operator and an empty set. Let $B$ be another alphabet. Show that each function $f : A \to B^*$ extends to a homomorphism $\varphi : A^* \to B^*$. (Hint: Letter for string substitution.)

2. The equational theory of semigroups is given by $E_1$:

\[
\begin{align*}
1 \cdot x_1 &= x_1, \\
x_1 \cdot 1 &= x_1, \\
x_1 \cdot (x_2 \cdot x_3) &= (x_1 \cdot x_2) \cdot x_3,
\end{align*}
\]

where $\Sigma = \{1, \cdot\}$. Determine the equivalence classes in $T(E_1)$ analogously to Example 1.8.

3. The equational theory of inf-semilattices is given by the axioms $E_2$:

\[
\begin{align*}
\top \land x_1 &= x_1, \\
x_1 \land \top &= x_1, \\
x_1 \land (x_2 \land x_3) &= (x_1 \land x_2) \land x_3, \\
x_1 \land x_2 &= x_2 \land x_1, \\
x_1 \land x_1 &= x_1,
\end{align*}
\]

where $\Sigma = \{\top, \land\}$. Repeat the previous exercise for this theory. Show that in a model $\mathcal{A}$ for the theory, $a \land^\mathcal{A} b$ is indeed the infimum of $a$ and $b$, under the partial order defined by $a \leq b \iff a = a \land^\mathcal{A} b$.

4. Try to find simple representatives of equivalence classes in $T_0(E)$ where $E$ is as in Remark 1.10.

5. Restricting the equational logic and putting more requirements on the axioms suggests a proof search strategy. Call an equational theory $E$ over $\Sigma$ instantiation closed if

(a) $s = t \in E$ implies $s^\sigma = t^\sigma \in E$ for each substitution $\sigma : \mathcal{X} \to \text{Ter}(\Sigma)$.

(b) $s = t \in E$ implies $t = s \in E$.

Let $\vdash_r$ denote the derivation relation which is as $\vdash_{eq}$ but where derivations are restricted to using only (ax.appl.), (cong), (refl) and (trans). Let $\vdash_d$ be the further restriction that (trans) is disallowed.

Consider an instantiation closed theory $E$. 
(i) Prove by induction on the height of proofs that for all terms \( s, t \)

\[ E \vdash_s s = t \implies E \vdash_s t = s. \]

(ii) Prove that for all terms \( s, t \) and all substitutions \( \sigma \)

\[ E \vdash_s s = t \implies E \vdash_s s^\sigma = t^\sigma. \]

(iii) Conclude that

\[ E \vdash_{\text{eq}} s = t \iff E \vdash_s s = t. \]

(iv) Define \( E \vdash^n_d s = t \) iff there are terms \( s_1, \ldots, s_{n-1} \) such that \( s_1 \equiv s \)

\[ E \vdash_d s_1 = s_2 \quad E \vdash_d s_2 = s_3 \quad \cdots \quad E \vdash_d s_{n-1} = t. \]

Prove that

\[ E \vdash_{\text{eq}} s = t \iff \text{for some } n \geq 1: E \vdash^n_d s = t. \]

Hint: transform the proofs so that transitivity applications appear at the end. Use transformations of the following kind, where \( D, D', D'' \), are proof trees.

\[
\frac{D}{s \equiv t} \quad \frac{D'}{a \equiv b} \quad \frac{D''}{b \equiv c} \quad \frac{D}{f(s, a) = f(t, c)} \quad (\text{trans}) \quad (\text{cong})
\]

\[
\frac{D}{s \equiv t} \quad \frac{D'}{a \equiv b} \quad \frac{D''}{b \equiv c} \quad \frac{D}{f(t, a) = f(t, c)} \quad (\text{refl}) \quad (\text{con}) \quad (\text{trans})
\]

### 1.3 Unification of terms

Unification is an important tool in term rewriting, automatic theorem proving, and is fundamental for logic programming (Prolog). Unification of terms amount to equation solving in the term algebra \( \text{Ter}(\Sigma) \).

**Example 1.11** Let \( \Sigma = \{ f, g \} \) with arities 2 and 1 respectively. Find a solution in \( \text{Ter}(\Sigma) \) to the equation

\[ f(x_1, g(f(x_2, x_1))) = f(g(x_2), x_3). \]
A solution: \( x_1 := g(x_2), x_3 := g(f(x_2, g(x_2))) \).

As in ordinary equation solving we are often interested in a general solution. Over the term algebra such a solution is called a most general unifier. Indeed, in the example above any other solution can be gotten from the one provided, by instantiating the variables.

As explained in Section 1.1 substitutions can be regarded as \( \Sigma \)-algebra homomorphisms \( \sigma : \text{Ter}(\Sigma) \to \text{Ter}(\Sigma) \) determined by their values on the set \( X \) of variables. A substitution that is given by a permutation of the variables is called a renaming substitution. Two substitutions \( \tau : \text{Ter}(\Sigma) \to \text{Ter}(\Sigma) \) and \( \sigma : \text{Ter}(\Sigma) \to \text{Ter}(\Sigma) \) may be composed \( \sigma \circ \tau \) as follows

\[
(\sigma \circ \tau)(t) = \sigma(\tau(t)) = (\tau^\sigma).
\]

We write \( \tau \sigma \) for \( \sigma \circ \tau \).

**Example 1.12** Let \( \Sigma = \{f, g\} \) with arities 2 and 1 respectively. Consider the substitutions \( \sigma = \{x_2 := f(x_1), x_3 := g(x_1)\} \) and \( \tau = \{x_1 := f(x_2, x_2)\} \). Then \( (\tau \sigma)(x_1) = \sigma(\tau(x_1)) = \sigma(f(x_2, x_2)) = f(\sigma(x_2), \sigma(x_2)) = f(\sigma(g(x_1), g(x_1)), (\tau \sigma)(x_2) = g(x_1) \) and \( (\tau \sigma)(x_3) = g(x_3) \). Hence

\[
\tau \sigma = \{x_1 := f(g(x_1), g(x_1)), x_2 := g(x_1), x_3 := g(x_3)\}.
\]

On the other hand, by a similar computation,

\[
\sigma \tau = \{x_1 := f(x_2, x_2), x_2 := g(f(x_2, x_2)), x_3 := g(x_3)\}. \quad \square
\]

Generalising this example we have for \( \sigma = \{x_{i_1} := t_1, \ldots, x_{i_n} := t_n\} \) and \( \tau = \{x_{i_1} := s_1, \ldots, x_{i_n} := s_n, x_{j_1} := r_1, \ldots, x_{j_m} := r_m\} \), where the indices \( i_1, \ldots, i_n, j_1, \ldots, j_m \) are all distinct, that

\[
\sigma \tau = \{x_{i_1} := t_{i_1}^r, \ldots, x_{i_n} := t_{i_n}^r, x_{j_1} := r_1, \ldots, x_{j_m} := r_m\}
\]

We say that one substitution \( \sigma \) is more general than another substitution \( \rho \) iff \( \rho = \sigma \tau \) for some substitution \( \tau \). In this case we write \( \sigma \leq \rho \).

**Exercise 1.13**

(i) Check that the relation \( \leq \) is reflexive and transitive.

(ii) Prove that if \( \sigma \leq \rho \) and \( \rho \leq \sigma \), then there is a renaming substitution \( \tau \) such that \( \rho = \sigma \tau \). \( \square \)
A unifier of a set of terms $\mathcal{T} = \{t_1, \ldots, t_n\}$ is substitution $\sigma$ which makes all these terms equal, i.e. $t_1^\sigma = \cdots = t_n^\sigma$. A unifier $\sigma$ of $\mathcal{T}$ is a most general unifier (mgu), if $\sigma \leq \rho$ for any unifier $\rho$ of $\mathcal{T}$. By Exercise 1.13 any two mgu’s $\sigma$ and $\sigma'$ of $\mathcal{T}$ are the same up to a renaming substitution (i.e. $\sigma = \sigma' \tau$ for some renaming substitution $\tau$).

Note that $F(s_1, \ldots, s_n)^\sigma = F(t_1, \ldots, t_n)^\sigma$ iff $s_i^\sigma = t_i^\sigma$ for all $i = 1, \ldots, n$. Hence in order to solve one equation in the term algebra, we may have to solve a system of equations.

The unification algorithm of Martelli-Montanari. The algorithm starts with a finite set of equations $G = \{s_1 = t_1, \ldots, s_n = t_n\}$, and outputs a most general unifier $\sigma$ for this set (regarded as an mgu of the set $\{F(s_1, \ldots, s_n), F(t_1, \ldots, t_n)\}$ where $F$ is a function symbol), if there is any unifier, or reports failure otherwise. The algorithm is non-deterministic and applies certain reduction rules to the finite sets and stops at the empty set ($\emptyset$), or with a failure (denoted #). Along the way the answer substitution $\sigma$ is built up. From a successful computation $G_1 \rightarrow G_2 \rightarrow_{\sigma_1} G_3 \rightarrow G_4 \rightarrow_{\sigma_2} G_5 \rightarrow G_6 \rightarrow \emptyset$.

we extract $\sigma = \sigma_1 \sigma_2$, the answer substitution. For a set $G = \{s_1 = t_1, \ldots, s_n = t_n\}$ we write $G^\sigma = \{s_1^\sigma = t_1^\sigma, \ldots, s_n^\sigma = t_n^\sigma\}$.

The Martelli-Montanari reduction rules are the following.

1. $G \cup \{F(t_1, \ldots, t_n) = F(s_1, \ldots, s_n)\} \rightarrow G \cup \{t_1 = s_1, \ldots, t_n = s_n\}$ provided $F(t_1, \ldots, t_n) = F(s_1, \ldots, s_n)$ is not an element of $G$. (“Function decomposition”)

2. $G \cup \{t = t\} \rightarrow G$ provided $t = t$ is not an element of $G$.

3. $G \cup \{t = x\} \rightarrow G \cup \{x = t\}$, provided $t$ is not a variable, and that $t = x$ is not an element of $G$.

4. $G \cup \{x = t\} \rightarrow_{\{x := t\}} G^{\{x := t\}}$, provided $x$ is a variable, $x$ does not occur in $t$ and that $x = t$ is not an element of $G$. (“Variable elimination”)

5. $G \cup \{F(t_1, \ldots, t_n) = H(s_1, \ldots, s_m)\} \rightarrow \#$, if $F$ and $H$ are different function symbols.

6. $G \cup \{x = t\} \rightarrow \#$, provided $x \neq t$ and $x$ occurs in $t$. (“Occur check”)

10
Example 1.14 We compute the mgu of $f(x_1, g(f(x_2, x_1)))$ and $f(g(x_2), x_3)$ using the algorithm.

\[
\{f(x_1, g(f(x_2, x_1))) = f(g(x_2), x_3)\} \implies \{x_1 = g(x_2), g(f(x_2, x_1)) = x_3\}
\implies \{x_1 := g(x_2)\} \implies \{x_3 = g(f(x_2, g(x_2)))\}
\]

The answer substitution is $\sigma = \{x_1 := g(x_2), x_3 := g(f(x_2, g(x_2)))\}$. □

Example 1.15 The terms $f(g(x_1), x_1)$ and $f(x_2, g(x_2))$ are not unifiable.

\[
\{f(g(x_1), x_1) = f(x_2, g(x_2))\} \implies \{g(x_1) = x_2, x_1 = g(x_2)\}
\implies \{x_1 := g(x_2)\} \implies \{g(x_2) = x_2\}
\implies \{x_2 = g(x_2)\}
\]

This computation fails by occur check, since $x_2$ occurs in $g(x_2))$. □

We state the following important result without proof:

**Theorem 1.16 (Unification Theorem)** A set of equations $G = \{s_1 = t_1, \ldots, s_n = t_n\}$ has an mgu iff it has some unifier. Moreover, if $G$ has an mgu, the Martelli-Montanari algorithm finds it, otherwise it stops and reports failure to find a unifier.

1.3.1 Pattern matching

Pattern matching may be regarded as a special case of unification: a variable free term $s$ is matched to the pattern term $t$ if there is a unifier $\sigma$ with

\[ s = s^\sigma = t^\sigma. \]

For a term $s$ containing variables, we may first replace each variable $x$ with a new constant $c_x$, and then match the modified term $s^*$ to $t$. The new constants occuring in the resulting unifier may then be restored to variables again.

Example 1.17 The term $f(0, g(2))$ is matched to $f(u, v)$ by $\sigma = \{u := 0, v := g(2)\}$.

The term $f(x, g(u))$ is matched to $f(u, v)$ by $\tau = \{u := x, v := g(u)\}$. The intermediate step is to consider the variable free term $f(c_x, g(c_u))$, and the unifier $\tau^* = \{u := c_x, v := g(c_u)\}$. Note that $\tau$ is not a unifier of $f(x, g(u))$ and $f(u, v)$.

However $f(x, g(u))$ cannot be matched to $f(g(u), v)$ since $c_x$ and $g(u)$ are not unifiable.
Exercises

1. Let $\Sigma = \{a, f, g, h, p, q\}$ where $a$ is a constant, $f, g$ has arity 1, $h, p$ has arity 2 and $q$ has arity 3. For each of the following pair of terms compute an mgu or show that no unifier exist.

   (a) $p(f(a), g(x)), p(y, y)$

   (b) $p(f(x), a), p(y, f(w))$

   (c) $p(x, x), p(y, f(y))$

   (d) $q(a, x, f(g(y))), q(z, h(z, w), f(w))$

   (e) $p(f(f(x)), h(g(x), f(a))), p(f(u), h(v, f(w))).$

2. In which cases (a) – (e) in Exercise 1 does the first term match the pattern of the second term?
2 Well-founded relations

A binary relation \((A, <)\) is well-founded if there is no infinite descending sequence
\[a_1 > a_2 > a_3 > \cdots\]
in \(A\).

**Example 2.1** The natural numbers \((\mathbb{N}, <)\) with the usual order is well-founded, while this is not the case for the integers \((\mathbb{Z}, <)\).

**Example 2.2** Consider \((\mathbb{N} \times \mathbb{N}, <')\) with the lexicographic order \((a, b) <' (c, d)\) iff \(a < c\) or \(a = c\) and \(b < d\). We have
\[(0, 0) <' (0, 1) <' \cdots <' (0, n) <' \cdots <' (1, 0) <' (1, 1) <' \cdots (2, 0) <' \cdots <' (m, 0).
\]
This relation is well-founded. For suppose \((a_{n+1}, b_{n+1}) <' (a_n, b_n)\) for all \(n\). Then the sequence \((a_n)\) is eventually constant from, say \(N\), and onwards. Hence \(b_{k+1} <' b_k\) for all \(k \geq N\), which is impossible. □

**Example 2.3** Let \(\Sigma\) be a signature and let \(\mathcal{X}\) be a nonempty set of variables. Order the set \(\text{Ter}(\Sigma, \mathcal{X})\) of terms over \(\Sigma\) and \(\mathcal{X}\) as follows
\[t \sqsubset s \iff t \neq s \text{ and } t \text{ is a subterm of } s.\]
If \(t \sqsubset s\), we say that \(t\) is a strict subterm of \(s\), or that it is structurally smaller than \(s\). We leave as an exercise to show that \((\text{Ter}(\Sigma, \mathcal{X}, \sqsubset)\) is a well-founded relation. Example: for \(\Sigma = \{0, f(\cdot), g(\cdot, \cdot)\}, \mathcal{X} = \{x, y, z, \ldots\}\) we have
\[x \sqsubset f(x) \quad y \not\sqsubset f(x) \quad f(y) \sqsubset g(f(f(y)), f(0)) \quad g(0, z) \sqsubset f(g(g(0, z), z))\] □
This kind of order relation is useful when proving termination of functional programs.

That a relation is well-founded is the same as saying that a certain induction principle is valid, so called Noetherian\(^1\) induction, or well-founded induction. Let \((A, <)\) be a binary relation. A subset \(S \subseteq A\) is progressive iff
\[(\forall a)[(\forall b < a)b \in S \Rightarrow a \in S].\]
Thus in a progressive set, if all the elements that lie before \(a\) are in the set, then also \(a\) is in the set. A binary relation \((A, <)\) is called inductive iff \(S = A\) whenever \(S \subseteq A\) is a progressive subset. What are the progressive subsets \(S\) of \((\mathbb{N}, <)\)? Clearly, there are no elements before 0, and hence trivially \(0 \in S\). Now suppose that \(\{0, 1, \ldots, n\} \subseteq S\). Then all elements before \(n + 1\) are in \(S\). Hence also \(n + 1 \in S\). By induction \(S = \mathbb{N}\). Above we just showed that \((\mathbb{N}, <)\) is inductive. In fact, we have

\(^1\)After Emmy Noether, a pioneer in abstract algebra.
**Theorem 2.4** A binary relation is well-founded iff it is inductive.

**Proof.** Suppose that \((A, <)\) is an inductive binary relation. Define the following subset of \(A\)

\[ S = \{ b \in A : \text{there is no infinite sequence } b > a_1 > a_2 > a_3 \cdots \}. \]

It is easily checked that \(S\) is progressive set. Hence \(S = A\), so \((A, <)\) is well-founded.

Now suppose that \((A, <)\) is not inductive. Hence there is a progressive set \(S \subset A\). Let \(x_0 \in A \setminus S\). Since \(S\) is progressive, there must be some \(x_1 < x_0\) such that \(x_1 \notin S\). But then again there must be some \(x_2 < x_1\) such that \(x_2 \notin S\). Proceeding in this way one constructs a sequence

\[ x_0 > x_1 > x_2 > \cdots \]

which shows that \((A, <)\) is not well-founded. \(\square\)

Let \((A, <)\) be a binary relation. The transitive closure \((A, <^+)\) of \((A, <)\) is defined by \(a <^+ b\) iff there is a sequence \(a_1 < \ldots < a_n, n \geq 1\), with \(a = a_1\) and \(b = a_n\). We leave the following as an easy exercise

**Proposition 2.5** Let \((A, <)\) be a binary relation. Then \((A, <)\) is well-founded iff \((A, <^+)\) is well-founded. \(\square\)

**Example 2.6** Let \(\Sigma\) be an arbitrary signature. For terms \(s, t \in \text{Ter}(\Sigma)\), define the immediate strict subterm relation \(s < t\) to hold if, and only if, \(t = f(s_1, \ldots, s_n)\) and \(s = s_i\) for some \(i\). Since terms are finite this is a well-founded relation. Then \(<^+\) is well-founded, and is the strict subterm relation \(\sqsubset\).

Suppose that \((A, <)\) is well-founded, \((B, <')\) a binary relation and \(f : B \to A\) a function such that, for all \(x\) and \(y\)

\[ x <' y \Rightarrow f(x) < f(y). \]

Then \((B, <')\) is well-founded. This fact can sometimes provide an easy proof that a relation is well-founded.

For instance consider Example 2.3. We know that \((\mathbb{N}, <)\) is well-founded. Let \(h : \text{Ter}(\Sigma, \mathbb{X}) \to \mathbb{N}\) be the height function where \(h(a) = 0\) for variables and constants \(a\), and for a function term of arity \(n \geq 1\)

\[ h(f(t_1, \ldots, t_n)) = 1 + \max(h(t_1), \ldots, h(t_n)). \]

Clearly \(t \sqsubset s\) implies \(h(t) < h(s)\). This shows that the strict subterm order is well-founded.
**Lexicographic orderings.** Let \((A, <_A)\) and \((B, <_B)\) be two binary relations. The *lexicographic combination* of these relations \((A \times B, <_{A,B})\) is defined as

\[
(x, y) <_{A,B} (u, v) \iff x <_A u \text{ or } x = u \text{ and } y <_B v.
\]

**Proposition 2.7** Let \((A, <_A)\) and \((B, <_B)\) be well-founded binary relations. Then their lexicographic combination \((A \times B, <_{A,B})\) is well-founded.

**Proof.** Analogous to Example 2.2. \(\Box\)

**Well-quasi-orders.** We introduce a notion related to that of a well-founded set. A binary relation \((A, R)\) is a *quasi-order* if it is reflexive and transitive. A quasi-order \((A, R)\) is a *well-quasi-order* if for every infinite sequence \(a_1, a_2, a_3, \ldots\) in \(A\) there are some \(m < n\) such that \(R(a_m, a_n)\).

**Example 2.8** \((\mathbb{N}, \leq)\) is a well-quasi-order. This is the case, since every infinite sequence in \(\mathbb{N}\) has a minimum.

More generally we have:

**Proposition 2.9** Let \((A, <)\) be a linear order. Define the relation

\[
x \leq y \iff \lnot y < x.
\]

Then \((A, <)\) is wellfounded iff \((A, \leq)\) is a well-quasi-order.

**Proof.** Suppose that \((A, <)\) is wellfounded. Let \(a_1, a_2, a_3, \ldots\) be an infinite sequence of elements of \(A\). Then it is impossible that \(a_{i+1} < a_i\) for all \(i\). Hence \(\lnot(a_{i+1} < a_i)\) for some \(i\). That is \(a_i \leq a_{i+1}\).

Conversely, assume that \((A, \leq)\) is a well-quasi-order. Suppose that \(a_1 > a_2 > a_3 > \cdots\) is an infinite, strictly decreasing sequence in \(A\). Then since \(\leq\) is a well-quasi-order, there are some \(m < n\), such that \(a_m \leq a_n\). By transtivity and linearity of \(<\) it follows that \(a_m = a_n\) — a contradiction. \(\Box\)

**Lemma 2.10** In any infinite sequence \(a_1, a_2, a_3, \ldots\) of natural numbers there is an infinite subsequence such that \(b_1 \leq b_2 \leq b_3 \leq \cdots\).

**Proof.** Let \(b_1\) be the first minimum (say \(a_{i_1}\)) of the sequence \(a_1, a_2, a_3, \ldots\). Let \(b_2 = a_{i_2}\) be the first minimum of the remaining sequence \(a_{i_1+1}, a_{i_1+2}, a_{i_1+3}, \ldots\). Let \(b_3 = a_{i_3}\) be the first minimum of the remaining sequence \(a_{i_2+1}, a_{i_2+2}, a_{i_2+3}, \ldots\) and so on. Clearly \(b_1, b_2, b_3, \ldots\) forms an increasing subsequence of the given sequence. \(\Box\)
Example 2.11 The relation \( P(x,y) \): \( x \) is a substring of a permutation of \( y \) is a quasi-order on \( L = \{0,1\}^* \). We have \( P(1010,010010) \) but \( \neg P(1011,010010) \).

It is more difficult to see that \( P \) is actually a well-quasi-order. Let \( s(a,x) \) denote the number of occurrences of \( a \) in \( x \). Note that \( P(x,y) \) iff \( s(0,x) \leq s(0,y) \) and \( s(1,x) \leq s(1,y) \). The relation is then expressed in terms of occurrence count. Suppose now that \( u_1,u_2,u_3,\ldots \) is a given sequence strings in \( L \). Consider the sequence \( s(0,u_1),s(0,u_2),s(0,u_3),\ldots \) of natural numbers. Then by Lemma 2.10 there is a subsequence \( v_1,v_2,v_3,\ldots \) of the given sequence such that

\[
s(0,v_1) \leq s(0,v_2) \leq s(0,v_3) \leq \cdots
\]

Since \( (\mathbb{N},\leq) \) is a well-quasi-order there is in this sequence some \( m < n \) such that \( s(1,v_m) \leq s(1,v_n) \). But then \( P(v_m,v_n) \) which was to be proven. \( \Box \)

This result can be generalised to arbitrary finite alphabets (See Exercises). Thus if you have an infinite row of books (which may arbitrary thick) there is always some book whose text may be obtained by cutting out letters from another book and rearranging them. Even more amazingly, you do not have to rearrange the letters:

**Proposition 2.12** Let \( \Sigma \) be a finite alphabet. Define the relation on the set \( \Sigma^* \) of strings:

\[
K(u,v) \iff u \text{ is obtained by removing zero or more symbols from } v.
\]

Then \( K \) is a well-quasi-order.

**Proof.** A sequence \( u_1,u_2,u_3,\ldots \) of strings in \( \Sigma^* \) is called bad if \( \neg K(u_m,u_n) \) for all \( m < n \). Suppose that \( K \) is not a well-quasi-order. Thus there is at least one bad sequence. In a bad sequence there are no empty strings, and any subsequence is obviously still bad. Let \( v_1 \) be a shortest string which is the first term of a bad sequence. Then let \( v_2 \) be a shortest string such that \( v_1,v_2 \) are the first two terms of a bad sequence. More generally, let \( v_n \) be a shortest string such that \( v_1,v_2,\ldots,v_n \) are the first \( n \) terms of a bad sequence. Each \( v_i \) is a non-empty string, so we may write \( v_i = a_iw_i \) where \( a_i \in \Sigma \). Since \( \Sigma \) is finite, some symbol occurs as initial symbol in infinitely many of the strings \( v_i \). Let \( k_1 \) be the least such that \( a_{k_1} \) occurs infinitely often as initial symbol. Suppose that \( k_1 < k_2 < k_3 < \ldots \) are the indices of strings that begin with \( a_{k_1} \). Then

\[
a_1w_1,a_2w_2,\ldots,a_{k_1-1}w_{k_1-1},w_{k_1},w_{k_2},w_{k_3},\ldots
\]

is a bad sequence, since all \( v_{k_i} \) begin with the same symbol \( a_{k_1} \) which is different from \( a_1,\ldots,a_{k_1-1} \). But now \( w_{k_1} \) is one symbol shorter than \( v_{k_1} \), contradicting the construction of \( v_{k_1} \).

Hence there are no bad sequences. \( \Box \)
Kruskal’s Theorem

This theorem is very useful proving termination of term rewriting systems. We refer to Dershowitz and Jouannaud (1990).

Let $T$ be the set of terms formed in the following way:

(a) $m \in T$ for any $m \in \mathbb{N}$,

(b) If $m \in \mathbb{N}$, and $t_1, \ldots, t_k \in T$ then $m(t_1, \ldots, t_k) \in T$.

Thus $3, 0(1,0(2)), 2(1,3(2,2,1(0)))$ are some examples of such terms.

Define the following quasi-order on $T$: $t \preceq s$ if $t$ can be obtained from $s$ by zero or more of the following operations

(a) replace $m(t_1, \ldots, t_n)$ by $t_i$ where $1 \leq i \leq n$

(b) replace $m(t_1, \ldots, t_n)$ by $p(t_1, \ldots, t_n)$ for $p < m$

(c) replace $m(t_1, \ldots, t_i, \ldots, t_n)$ by $q(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$ for $1 \leq i \leq n$ and $q \leq m$.

Example 2.13 We have $0(1,0(2)) \preceq 2(1,0(2)) \preceq 2(1,3(2)) \preceq 2(1,3(2,2)) \preceq 2(1,3(2,2,1)) \preceq 2(1,3(2,2,1(0)))$. But $1(1,1) \not\preceq 1(1,0)$.

Theorem 2.14 (Kruskal) $\preceq$ is a well-quasi-order on $T$.

The proof is difficult and beyond the scope of this course. However, as much can be said that it uses the technique of minimal bad sequences illustrated in Lemma 2.10 and Proposition 2.12.

Exercises

1. Show that the following program $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ terminates by using a lexicographic combination

   $f(0, y) = y$

   $f(S(x), 0) = S(x)$

   $f(S(x), S(y)) = S(f(x, f(x, y)))$.

2. Prove Proposition 2.5.

3. Show that $\sqsubset$ is the transitive closure of the immediate subterm relation on $\text{Ter}(\Sigma, X)$. 
4. Extend Example 2.11 to strings over any finite alphabet. (What does Proposition 2.12 say here?)

5. Prove that the substring relation over \{0, 1\}^* is not a well-quasi-order.

6. Find all \( t \in T \) such that \( t \preceq 2(1, 1(2, 1(0))) \).

### 2.1 Abstract reduction systems

An Abstract Reduction System (ARS) is a set \( A \) together with a binary relation \( \rightarrow \). Further on we will mostly be interested in the case where \( A \) is a set of terms and \( \rightarrow \) is a one-step computation, or reduction, relation. However we treat the general case first, so \( (A, \rightarrow) \) could be any directed graph, finite or infinite.

An element \( a \) in \( A \) of an ARS \( (A, \rightarrow) \) is said to be a normal form, if there is no \( b \in A \) such that \( a \rightarrow b \). (Intuitively \( a \) cannot be computed further, and can be considered as the value of a computation.)

**Example 2.15** Let \( A = \{0, 1, 2, 3\} \) and \( \rightarrow = \{(1, 0), (1, 2), (2, 1), (2, 3)\} \). (Draw the graph of this ARS!) It is easy to see that the elements of normal form are exactly 0 and 3.

**Example 2.16** The ARS given by \( A_2 = \{0, 1\} \) and \( \rightarrow = \{(1, 0), (0, 1)\} \) has no elements of normal form.

Let \( (A, \rightarrow) \) be an ARS. Denote by \( \rightarrow^+ \) the reflexive and transitive closure of \( \rightarrow \), that is, \( a \rightarrow b \) holds iff there is a sequence \( a = a_1, \ldots, a_n = b, n \geq 1 \), such that

\[
a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_n.
\]

Write \( a \rightarrow^+ b \) if this holds for a sequence where \( n \geq 2 \). An ARS \( (A, \rightarrow) \) is weakly normalizing (WN) if for every \( a \in A \) there is some normal form \( b \in B \) with \( a \rightarrow b \). It is easily checked that the ARS of Example 2.15 is weakly normalizing. Note however that 1 \( \rightarrow 0 \) and 1 \( \rightarrow 3 \) so that 1 has two distinct normal forms.

Two elements \( a \) and \( b \) of an ARS \( (A, \rightarrow) \) are said to be convergent (in symbols: \( a \downarrow b \)) if there is some \( c \) such that \( a \rightarrow c \) and \( b \rightarrow c \). An ARS \( (A, \rightarrow) \) is confluent or Church-Rosser (CR) if \( b \downarrow c \) for any \( a, b, c \in A \) such that \( a \rightarrow b \) and \( a \rightarrow c \). The following simple result shows the importance of this property.

**Proposition 2.17** Let \( (A, \rightarrow) \) be a confluent, weakly normalizing ARS. Then every element of \( A \) has a unique normal form.
Proof. Suppose that $b$ and $c$ are normal forms and $a \rightarrow b$ and $a \rightarrow c$. By confluence, for some $d \in A$ with $b \rightarrow d$ and $c \rightarrow d$. Since $b$ is normal, $b = d$ and likewise $c = d$. Hence $b = c$. □

An ARS $(A, \rightarrow)$, where there are no infinite sequences $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \cdots$ is called strongly normalizing (SN), i.e. $(A, \leftarrow)$ is a wellfounded relation. Clearly, in this case any strategy of performing the reductions will lead to a normal form. The ARS of Example 2.15 is does not have this property since there is the sequence $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow \cdots$.

Example 2.18 The ARS given by $A = \{0, 1, 2, 3\}$ and $\rightarrow = \{(1, 0), (1, 2), (2, 3)\}$ is strongly normalizing but not confluent.

The following theorem is often useful when proving confluence. An ARS $(A, \rightarrow)$ is weakly confluent or Weakly Church-Rosser (WCR) if $b \downarrow c$ for any $a, b, c \in A$ such that $a \rightarrow b$ and $a \rightarrow c$. (Note the one-step computation relations from $a$.)

Theorem 2.19 (Newman’s lemma) A weakly confluent, strongly normalizing ARS is confluent.

Proof. Let $(A, \rightarrow)$ be an ARS. That it is confluent is equivalent to $P(u)$ for all $u$, where

$$P(u) \iff_{\text{def}} (\forall x, y)[u \rightarrow x \land u \rightarrow y \Rightarrow x \downarrow y]$$

Since the ARS is strongly normalizing, we can prove $(\forall u)P(u)$ by Noetherian induction. For this it suffices to show that $S = \{u \in A : P(u)\}$ is a progressive set, i.e.

$$(\forall u)[(\forall t) (u \rightarrow t \Rightarrow P(t)) \Rightarrow P(u)].$$

So assume that $u \in A$ is arbitrary, and as induction hypothesis $(\forall t) (u \rightarrow t \Rightarrow P(t))$. In case $u$ is normal, we are done. Otherwise, suppose that $u \rightarrow b \rightarrow x$ and $u \rightarrow c \rightarrow y$. By weak confluence there is some $d$ such that $b \rightarrow d$ and $c \rightarrow d$. By the induction hypothesis $P(b)$, so there is a $z$ with $x \rightarrow z$ and $d \rightarrow z$. By transitivity, $c \rightarrow z$. Using the induction hypothesis again, $P(c)$ holds, so there is some $v$ with $z \rightarrow v$ and $y \rightarrow v$. Thus by transitivity, $x \rightarrow v$. The induction step is finished. □
3 Term rewriting systems

We shall be brief and sketchy on this subject. We refer to Baader and Nipkow (1999) or to Klop (1992) for a full account of the basic theory.

A term rewriting system (TRS) is essentially a way of assigning directions to the equations of an equational theory $E$, and then applying the equations only in the prescribed directions. In some circumstances a TRS can be devised for $E$ so that it can be decided whether $E \vdash_{eq} s = t$ holds by making “mindless” applications of the directed equations to the terms $s$ and $t$ respectively and when no further applications are possible check whether the end results are the same. We shall explain what “mindless application” means below.

First we give an example of a TRS. Consider the equational theory of semi-groups (Exercise 1.2). The language is $\Sigma = \{1, \cdot\}$. The equations

\[
1 \cdot x = x \\
x \cdot 1 = x \\
x \cdot (y \cdot z) = (x \cdot y) \cdot z
\]

may be given the natural directions

\[
1 \cdot x \rightarrow x \\
x \cdot 1 \rightarrow x \\
x \cdot (y \cdot z) \rightarrow (x \cdot y) \cdot z
\]

These are called rewrite rules.

Consider the following applications of the directed equations, so called reductions. We underline the subterms to which the rewrite rules have been applied

\[
1 \cdot (x_2 \cdot x_3) \xrightarrow{(3)} (1 \cdot x_2) \cdot x_3 \xrightarrow{(1)} x_2 \cdot x_3 \\
1 \cdot (x_2 \cdot x_3) \xrightarrow{(1)} x_2 \cdot x_3. \\
(x_2 \cdot 1) \cdot x_3 \xrightarrow{(2)} x_2 \cdot x_3.
\]

It is not possible to apply any further rules to $x_2 \cdot x_3$ since $x_2$ and $x_3$ are variables and stand for arbitrary objects.

Let $\Sigma$ be an arbitrary signature. A rewrite rule over $\Sigma$ is a pair $(s, t) \in \text{Ter}(\Sigma)^2$, written $s \rightarrow t$, so that $s$ is not a variable and $\text{FV}(t) \subseteq \text{FV}(s)$. A term rewriting system over $\Sigma$ is a finite set of rewrite rules over $\Sigma$. 

Example 3.1 \( \Sigma_1 = \{1, \cdot\} \) and \( R_1 = \{1 \cdot x_1 \rightarrow x_1, x_1 \cdot 1 \rightarrow x_1, x_1 \cdot (x_2 \cdot x_3) \rightarrow (x_1 \cdot x_2) \cdot x_3\} \) is the TRS for semigroups above.

We allow variable names \( x, y, z, u, v, w \) as well as the official variables \( x_1, x_2, x_3, \ldots \) of the set \( \mathbb{X} \).

Example 3.2 A term rewriting system for addition and multiplication: \( \Sigma_2 = \{0, +, \cdot, s\} \) and \( R_2 \) consists of the following rules

\[
\begin{align*}
x + 0 & \rightarrow x \\
x + s(y) & \rightarrow s(x + y) \\
x \cdot 0 & \rightarrow 0 \\
x \cdot s(y) & \rightarrow x \cdot y + x
\end{align*}
\]

Let \( R \) be a TRS over \( \Sigma \). For two terms \( t_1, t_2 \in \text{Ter}(\Sigma) \) we say that \( t_2 \) has been obtained by one-step reduction from \( t_1 \) using \( R \), in symbols

\( t_1 \rightarrow_R t_2 \)

if \( t_1 = C(z:=s^\sigma), t_2 = C(z:=t^\sigma) \) for some rule \((s,t) \in R\), some variable \( z \), some substitution \( \sigma \) and some term \( C \) with exactly one occurrence of \( z \).

Example 3.3 We have the one-step computation

\[
u \cdot ((v \cdot w) \cdot 1) \rightarrow_{R_1} u \cdot (v \cdot w)
\]

taking \( C = u \cdot z \) and the \( R_1 \)-rule \( x_1 \cdot 1 \rightarrow x_1 \) of \( R_1 \), with the substitution \( \sigma = \{x_1 := u \cdot v\} \). Moreover we have the one-step computation

\[
s(s(0)) + s(s(0)) \rightarrow_{R_2} s(s(s(0)) + s(0))
\]

by taking \( C = z \), and the \( R_2 \)-rule \( x + s(y) \rightarrow s(x + y) \) with the substitution \( \sigma = \{x := s(s(0)), y := s(0)\} \).

We write \( \rightarrow^* \) for the reflexive and transitive closure of \( \rightarrow \). We say that \( s \) reduces to \( t \) if \( s \rightarrow^*_R t \).

We call a term \( t \in \text{Ter}(\Sigma) \) is called normal with respect to \( R \) if there is no term \( s \) such that \( t \rightarrow_R s \).

Note that variables are normal with respect to any TRS, since they can never be the left-hand side of a rule.
Example 3.4 For TRS $R_1$: 1 is normal. Each expression of the form
\[(\cdots((x_{i_1} \cdot x_{i_2}) \cdot x_{i_3}) \cdots x_{i_n})\]
where $n \geq 1$, is normal. In fact these are all the normal terms.

Example 3.5 For TRS $R_2$: The numerals 0, $s(0), s(s(0)), \ldots$ are normal. $x_1 + x_2$ and $x_1 \cdot x_2$ are normal. (Exercise: determine all normal terms)

A TRS $R$ is said to be confluent if for any terms $r, s_1, s_2$ with $r \rightarrow^* s_1$ and $r \rightarrow^* s_2$ there is a term $t$ such that $s_1 \rightarrow^* t$ and $s_2 \rightarrow^* t$. It is weakly confluent if for any terms $r, s_1, s_2$ with $r \rightarrow s_1$ and $r \rightarrow s_2$ there is a term $t$ such that $s_1 \rightarrow^* t$ and $s_2 \rightarrow^* t$.

We note that if a term $s$ is normal and $s \rightarrow^* t$, then $s = t$. Using this observation have the following simple but important result.

Lemma 3.6 If $R$ is a confluent TRS, then normal forms are unique if they exist, i.e. for any term $r$ and normal terms $s_1, s_2$ with $r \rightarrow^* s_1$ and $r \rightarrow^* s_2$, it holds that $s_1 = s_2$.

A TRS $R$ is weakly normalising if any term reduces to a normal term. It is strongly normalising if there are is no infinite sequence of terms such that
\[t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_n \rightarrow \cdots\]

For a strongly normalising TRS any sequence of choices of subterms and applicable rules will thus eventually lead to a normal term. E.g. any “mindless” application of the directed equation will give the result.

Moreover, this means that any strongly normalising TRS is weakly normalising.

Example 3.7 We consider here some rather degenerate TRS with only constants, to provide simple counter examples. These are in fact ARS (see previous chapter).

Let $\Sigma_3 = \{0, 1, 2, 3\}$. Consider TRSs given by the following rules over $\Sigma_3$.

$R_{3,1} = \{1 \rightarrow 0, 1 \rightarrow 2\}$ is strongly normalising but not confluent.

$R_{3,2} = \{1 \rightarrow 0, 1 \rightarrow 2, 2 \rightarrow 1, 2 \rightarrow 3\}$ is weakly normalising but not strongly normalising.

$R_{3,3} = \{1 \rightarrow 0, 1 \rightarrow 2, 2 \rightarrow 1\}$ is weakly normalising and confluent.

$R_{3,4} = \{1 \rightarrow 0, 1 \rightarrow 2, 0 \rightarrow 2 \rightarrow 2\}$ is neither weakly normalising nor confluent.

Theorem 3.8 (Newman’s Lemma) Any strongly normalizing, weakly confluent TRS is confluent.
Proof. This follows by Newman’s Lemma 2.19 for abstract reduction systems (ARS). □

For a TRS \( R \) let \( =_R \) be the reflexive, symmetric and transitive closure of the one-step rewrite relation \( \rightarrow_R \). Thus \( s =_R t \) if and only if \( s \) can be gotten from \( t \) by a series of one-step reductions possibly applying certain of them backwards.

A TRS is complete if it is confluent and strongly normalising. The importance of complete TRSs is given by the following result.

**Theorem 3.9** Let \( R \) be a complete TRS. Then the relation \( s =_R t \) is decidable.

Proof. We note that by the confluence property \( s =_R t \) is equivalent to the existence of some \( r \) with \( s \rightarrow^*_R r \) and \( t \rightarrow^*_R r \). To decide the equality \( s =_R t \) we compute normal forms \( s', t' \) with \( s \rightarrow^*_R s' \) and \( t \rightarrow^*_R t' \) using the strong normalisation property and that \( R \) is finite. If \( s' = t' \) then equality holds. Suppose that \( s' \neq t' \) but that \( s =_R t \) still holds. Then for some \( r \) with \( s \rightarrow^*_R r \) and \( t \rightarrow^*_R r \). Using confluence we find \( s'' \) and \( t'' \) with

\[
\begin{align*}
s' & \rightarrow^*_R s'' \\
r & \rightarrow^*_R s'' \\
r & \rightarrow^*_R t'' \\
t' & \rightarrow^*_R t''
\end{align*}
\]

Now since \( s' \) and \( t' \) are normal we have \( s' = s'' \) and \( t' = t'' \) by confluence. But this means that \( r \) reduces to two different normal forms, which is impossible. Hence actually, \( s =_R t \) is false. □

Let \( E \) be an equational theory over \( \Sigma \). Let \( R \) be a TRS over \( \Sigma \). We say that \( R \) is a TRS for \( E \) if for all \( s, t \)

\[
E \vdash_{eq} s = t \iff s =_R t.
\]

A main result is now:

**Corollary 3.10** Suppose \( E \) be an equational theory over \( \Sigma \). Let \( R \) be a complete TRS for \( E \). Then the provability relation

\[
E \vdash_{eq} s = t
\]

is decidable.
Proof. This follows since $E \vdash_{eq} s = t$ is equivalent to $s =_R t$, which is decidable. □

A complete TRS does not always exists for a given equational theory $E$. However one can sometimes use the Knuth-Bendix completion procedure to obtain a complete TRS. Here is a motivating example:

Example. (After Klop 1992) A minimal equational theory for groups is given by the equations

\[
\begin{align*}
1 \cdot x &= x \\
x^{-1} \cdot x &= 1 \\
(x \cdot y) \cdot z &= x \cdot (y \cdot z).
\end{align*}
\]

Going from complex to simple expression these equations can naturally be oriented as

\[
\begin{align*}
1 \cdot x &\rightarrow x \\
x^{-1} \cdot x &\rightarrow 1 \\
(x \cdot y) \cdot z &\rightarrow x \cdot (y \cdot z).
\end{align*}
\]

which gives a TRS $R$. This system is not confluent, as we have

\[
(x^{-1} \cdot x) \cdot z \rightarrow_2 1 \cdot z \rightarrow_1 z
\]

using the second and first rule, on the one hand. On the other hand

\[
(x^{-1} \cdot x) \cdot z \rightarrow_3 x^{-1} \cdot (x \cdot z)
\]

using the third rule. Both $x^{-1} \cdot (x \cdot z)$ and $z$ are normal with respect to $R$, and they are obviously not syntactically equal. Thus confluence fail. Adding a fourth rule

\[
x^{-1} \cdot (x \cdot z) \rightarrow z
\]

to $R$ solves this confluence problem locally, but introduces new ones. Now we have the overlapping reduction possibilities

\[
(y^{-1})^{-1} \cdot (y^{-1} \cdot y) \rightarrow_4 y
\]

and

\[
(y^{-1})^{-1} \cdot (y^{-1} \cdot y) \rightarrow_2 (y^{-1})^{-1} \cdot 1.
\]

But $y$ and $(y^{-1})^{-1} \cdot 1$ are normal also with respect to the extended rule set, so confluence fails again.
The problem is here that there are overlapping rewrite possibilities that cannot be computed to the same expressions later. This is a so-called critical pair in the system $R$, which cannot be computed to the same expression. A general definition is the following.

If for a pair of rewrite rules $\alpha \rightarrow \beta$ and $\gamma \rightarrow \delta$ it holds that $\alpha$ can be unified with a subterm of $\gamma$, which is not a variable, then we have an overlap of these rules. Suppose that $\gamma = C(z = t)$ where $z$ is a variable that occurs exactly once in $C$, but not at all in $\alpha, \beta, \gamma, \delta$ and that $t$ is not a variable. Moreover suppose that $\alpha$ and $t$ has a most general unifier $\sigma$. We may assume $z^\sigma = z$. Then we have

$$\gamma^\sigma = C(z = t)^\sigma \rightarrow C^\sigma(z = t^\sigma)$$

and also

$$\gamma^\sigma \rightarrow \delta^\sigma.$$

The pair $(C^\sigma(z = t^\sigma), \delta^\sigma)$ is called a critical pair. It is convergent if for some $r$, $C^\sigma(z = t^\sigma) \rightarrow^* r$ and $\delta^\sigma \rightarrow^* r$.

**Theorem 3.11** (Knuth-Bendix 1970) A TRS is weakly confluent iff all its critical pairs are convergent. $\square$

**Corollary 3.12** (Knuth-Bendix 1970) A strongly normalizing TRS is confluent iff all its critical pairs are convergent.

**Proof.** By the above using Newman’s Lemma. $\square$

We say that a wellfounded partial order $<$ on $\text{Ter}(\Sigma)$ is a reduction order if

(a) $s^\sigma < t^\sigma$ whenever $s < t$ and $\sigma$ is a substitution,

(b) $C(z = s) < C(z = t)$ whenever $s < t$ and $C$ is a term with exactly one occurrence of $z$.

**Theorem 3.13** A TRS $R$ is strongly normalizing iff there is reduction order $>$ such that $\alpha > \beta$, whenever for each rule $\alpha \rightarrow \beta$ in $R$. $\square$

**Proof.** As for $(\Leftarrow)$ consider any infinite reduction sequence

$$u_1 \rightarrow_R u_2 \rightarrow_R u_3 \rightarrow_R \cdots$$

(2)

Now $u_n \rightarrow_R u_{n+1}$ means that

$$u_n = C(z = s^\sigma) \quad u_{n+1} = C(z = t^\sigma)$$

25
for some rule \((s, t) \in R\), some variable \(z\), some substitution \(\sigma\) and some term \(C\) with exactly one occurrence of \(z\). We \(s > t\) since \(s \rightarrow t\) is a rule. Moreover since \(>\) is a reduction order \(s^\sigma > t^\sigma\) by property (a) and then by property (b) \(C[z:=s^\sigma] > C[z:=t^\sigma]\). Hence \(u_n > u_{n+1}\). This hold for any \(n\). Hence \(<\) is not a well-order contrary to the assumption. We conclude that no infinite reduction sequence as in (2) exists. Hence \(R\) is strongly normalizing.

For \((\Rightarrow)\) see Klop (1992). \(\Box\)

The Knuth-Bendix completion procedure (see Klop 1992) takes \(E\) an equational theory over \(\Sigma\) and a reduction order \(<\) on \(\text{Ter}(\Sigma)\). It may then produce a complete TRS for \(E\), or report that that it is not possible to orient the equations of \(E\), or it may go on forever searching for a TRS. The method is to systematical search for critical pairs and add new rules trying to achieve confluence.

An obstacle for the existence of a complete TRS of a given equational theory is the presence of commutativity axioms like

\[ x \cdot y = y \cdot x. \]

They are in general impossible to orient.

A particularly well behaved class of TRSs are the orthogonal ones. A rule \(\alpha \rightarrow \beta\) is called left linear if no variable in \(\alpha\) occur more than once. For instance

\[ x \cdot (y \cdot z) \rightarrow (x \cdot y) \cdot z \]

and

\[ x \cdot (y + z) \rightarrow x \cdot y + x \cdot y \]

are left linear, while

\[ x \cdot x^{-1} \rightarrow 1 \]

is not. A TRS \(R\) is orthogonal if each rule is left linear and there are no critical pairs.

The TRS of Example 3.2 for addition and multiplication

\[
\begin{align*}
x + 0 & \rightarrow x \\
x + s(y) & \rightarrow s(x + y) \\
x \cdot 0 & \rightarrow 0 \\
x \cdot s(y) & \rightarrow x \cdot y + x
\end{align*}
\]

is clearly left linear. By systematically checking the left-hand sides for critical pairs we can conclude that it is also orthogonal. For instance: \(x + 0\) does not unify with any non-variable subterm of the left-hand sides of the other rules: \(x + s(y)\)
(subterms: $x + s(y)$, $x$, $s(y)$, $y$) $x \cdot 0$ (subterms: $x \cdot 0$, $x$, $0$) or $x \cdot s(y)$ (subterms: $x \cdot s(y)$, $x$, $s(y)$, $y$). Similarly we check the other left-hand sides against the remaining left-hand sides.

An fundamental result is the following

**Theorem 3.14** Any orthogonal TRS is confluent. □

**Exercises**

1. Investigate whether the TRS

$$x \cdot (y + z) \rightarrow x \cdot y + x \cdot z$$
$$\ (y + z) \cdot x \rightarrow y \cdot x + z \cdot x$$
$$\ (x \cdot y) \cdot z \rightarrow x \cdot (y \cdot z)$$

is confluent. Can you identify the normal terms (assuming the signature has just the operations $+$ and $\cdot$)? What about if we also add the rule

$$(x + y) + z \rightarrow x + (y + z)$$

2. Generalizing Example 3.2, show that the primitive recursive functions can be presented as an orthogonal term rewriting system.

3. Let $\Sigma = \{\cdot, k, s\}$ be a signature where $\cdot$ is a binary operation, and $k$ and $s$ are constants. Show that the TRS

$$\ (k \cdot x) \cdot y \rightarrow x$$
$$\ ((s \cdot x) \cdot y) \cdot z \rightarrow (x \cdot z) \cdot (y \cdot z)$$

is orthogonal. Show that it is not strongly normalizing. (Hint 1: there is a closed normal term $\omega = (s \cdot v) \cdot v$ with no more than six occurrences of $\cdot$ such that $\omega \cdot \omega \rightarrow^+ \omega \cdot \omega$. Tricky, but one can try them all! Hint 2: The TRS is actually a version of so-called combinatory logic. It is easy if you already know about the standard examples in this theory.)

4. (Dershowitz and J.P. Jouannaud 1990, p. 271) The following TRS is strongly normalizing, but cannot be proven to be so in Peano Arithmetic

$$h(z, e(x)) \rightarrow h(c(z), d(z, x))$$
$$d(z, g(0, 0)) \rightarrow e(0)$$
$$d(z, g(x, y)) \rightarrow g(e(x), d(z, x))$$
$$d(c(z), g(g(x, y), 0)) \rightarrow g(d(c(z), g(x, y)), d(z, g(x, y)))$$
$$g(e(x), e(y)) \rightarrow e(g(x, y))$$
References


