
1

Berekeningsmodellen

Hans Zantema

Voorjaar 2018, Termherschrijven

Donderdag 26 april, college:

Termherschrijven (TRS)

Leertaak: Reductie in TRS

Woensdag 9 mei: Responsiecollege

Donderdag 17 mei, college:

Normalisatie in TRS

Leertaak: Normalisatie in TRS

Woensdag 23 mei: Responsiecollege

Donderdag 24 mei, college:

Confluentie in TRS

Leertaak: Confluentie in TRS

Woensdag 30 mei: Responsiecollege

$+$ applied to natural numbers should yield a natural number after application of these basic rules.

Here natural numbers are defined to be closed terms composed from the constant 0 and the unary symbol s .

Hence we need rules by which every closed term containing the symbol $+$ can be rewritten to a closed term not containing $+$.

One way to do so is:

$$0 + x = x$$

$$s(x) + y = s(x + y)$$

2

Equational reasoning

We will give a minimal description of natural numbers in which $2 + 2 = 4$ makes sense and can be proved automatically.

Natural numbers:

$$0, s(0), s(s(0)), s(s(s(0))), \dots$$

These are the closed terms composed from the constant 0 and the unary symbol s .

Here a term is called **closed** if it does not contain variables.

We want to show that

$$s(s(0)) + s(s(0)) = s(s(s(s(0))))$$

Here $+$ is a binary operator written in infix notation.

3

This claim only holds if we have some basic rules for $+$:

4

What is the meaning of such rules?

- For variables (here: x, y) arbitrary terms may be substituted.
- These rules may be applied on any sub-term of a term that has to be rewritten.

In case the rules are only allowed to be applied from left to right we write an arrow \rightarrow instead of $=$.

The rules are called **rewrite rules**.

A set of such rewrite rules is called a **term rewrite system (TRS)**.

5

In order to define this more precisely, first we define **terms** and **substitution**.

A set Σ of function symbols is called a signature.

Function symbols have an **arity** = $0, 1, 2, 3, \dots$, indicating the number of arguments it expects.

A function symbol of arity 0 is also called a **constant**.

In our example we have

- the function symbol s of arity 1,
- the function symbol $+$ of arity 2,
- the constant 0 .

6

We inductively define:

A **term** is

- a variable, or
- a function symbol of arity n applied on n terms.

The default notation is **prefix**, i.e., the symbol f applied on terms t_1, \dots, t_n is written as $f(t_1, \dots, t_n)$.

For some symbols (in our case $+$) an infix notation is more standard, however, this requires some extra rules of how to deal with parentheses and priorities.

For a constant c we also write c for the corresponding term rather than $c()$.

For a signature Σ and a set \mathcal{X} of variables the corresponding set of terms is denoted by $\mathcal{T}(\Sigma, \mathcal{X})$.

7

A **substitution** is a map from variables to terms.

A substitution σ can be extended to arbitrary terms by inductively defining:

$$x\sigma = \sigma(x)$$

for every variable x and

$$f(t_1, \dots, t_n)\sigma = f(t_1\sigma, \dots, t_n\sigma)$$

for every function symbol f .

So $t\sigma$ is obtained from t by replacing every variable x in t by $\sigma(x)$.

For instance, if $\sigma(x) = y$ and $\sigma(y) = g(x)$ then

$$h(f(x), y)\sigma = h(f(y), g(x))$$

8

Definition:

A **term rewrite system** (TRS) R over a signature Σ is a subset of $\mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$.

An element $(\ell, r) \in R$ is called a **rule** and is usually written as $\ell \rightarrow r$ instead of (ℓ, r) .

ℓ is called the **left hand side** and r is called the **right hand side** of the rule.

The rewrite relation \rightarrow_R is defined to be the smallest relation $\rightarrow_R \subseteq \mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ satisfying:

- $\ell\sigma \rightarrow_R r\sigma$ for every $\ell \rightarrow r$ in R and every substitution σ , and
- if $t_j \rightarrow_R u_j$ and $t_i = u_i$ for every $i \neq j$, then $f(t_1, \dots, t_n) \rightarrow_R f(u_1, \dots, u_n)$.

9

This last property causes that application of rules is allowed on subterms.

For instance, if the TRS R consists of the rules

$$+(0, x) \rightarrow x, \quad +(s(x), y) \rightarrow s(+(x, y))$$

(the same as before, now in prefix notation) then indeed $2 + 2 = 4$ holds:

$$+(s(s(0)), s(s(0))) \rightarrow_R s(\underbrace{+(s(0), s(s(0)))})$$

$$\rightarrow_R s(\underbrace{s(+(0, s(s(0))))})$$

$$\rightarrow_R s(s(s(s(0))))$$

10

A term t is called a **normal form** if no u exists satisfying $t \rightarrow_R u$.

Computation
=
rewrite to normal form
=
apply rewriting as long as possible

So in our example rewriting to normal form of the term $2+2$ represented by $+(s(s(0)), s(s(0)))$ yields the term 4 represented by $s(s(s(s(0))))$.

A term t is called a **normal form of** a term u if t is a normal form and u rewrites to t in zero or more steps.

11

A rewriting sequence is also called a **reduction**; it can be infinite, unfinished, or end in a normal form.

Rewriting to normal form is the basic formalism in several kinds of computation.

In particular, it is the underlying formalism for both semantics and implementation of **functional programming**, in which the function definitions are interpreted as rewrite rules.

12

Example:

$\text{rev}(\text{nil}) = \text{nil}$
 $\text{rev}(a : x) = \text{conc}(\text{rev}(x), a : \text{nil})$
 $\text{conc}(\text{nil}, x) = x$
 $\text{conc}(a : x, y) = a : \text{conc}(x, y)$

Here a, x, y are variables, and $=$ corresponds to \rightarrow in rewrite rules.

Then we have a reduction to normal form

$\text{rev}(1:2:\text{nil}) \rightarrow$
 $\text{conc}(\text{rev}(2:\text{nil}), 1:\text{nil}) \rightarrow$
 $\text{conc}(\text{conc}(\text{rev}(\text{nil}), 2:\text{nil}), 1:\text{nil}) \rightarrow$
 $\text{conc}(\text{conc}(\text{nil}, 2:\text{nil}), 1:\text{nil}) \rightarrow$

$\text{conc}(2:\text{nil}, 1:\text{nil}) \rightarrow$
 $2:\text{conc}(\text{nil}, 1:\text{nil}) \rightarrow$
 $2:1:\text{nil}$

13

Without extra requirements a term can have no normal form, or more than one normal form.

For instance, with respect to $f(x) \rightarrow f(x)$ the term $f(a)$ does not have a normal form.

For instance, with respect to $f(f(x)) \rightarrow a$ the term $f(f(f(a)))$ has two normal forms a and $f(a)$.

Now we investigate some nice properties forcing that every term has exactly one normal form.

14

Nice properties:

- R is **terminating** (= strongly normalizing, SN):

no infinite sequence of terms
 t_1, t_2, t_3, \dots exists such that $t_i \rightarrow_R t_{i+1}$ for all i .

- R is **weakly normalizing** (WN) if every term has at least one normal form.

- R is **confluent** (= Church-Rosser, CR):

if $t \rightarrow_R^* u$ and $t \rightarrow_R^* v$ then a term w exists satisfying $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$.

- R is **locally confluent** (= weak Church-Rosser, WCR):

if $t \rightarrow_R u$ and $t \rightarrow_R v$ then a term w exists satisfying $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$.

Here \rightarrow_R^* is the reflexive transitive closure of \rightarrow_R , i.e., $t \rightarrow_R^* u$ if and only if t can be rewritten to u in zero or more steps.

15

Property

If a TRS is terminating, then every term has at least one normal form.

Proof: rewriting as long as possible does not go on forever due to termination.

So it ends in a normal form.

The converse is not true: the TRS over the two constants a, b consisting of the two rules $a \rightarrow a$ and $a \rightarrow b$ is weakly normalizing since the two terms a and b both have b as a normal form, but it is not terminating due to

$$a \rightarrow a \rightarrow a \rightarrow a \rightarrow \dots$$

16

Property

If a TRS is confluent, then every term has at most one normal form.

Proof: Assume t has two normal forms u, u'

Then by confluence there is a v such that $u \rightarrow_R^* v$ and $u' \rightarrow_R^* v$.

Since u, u' are normal forms we have $u = v = u'$.

17

Termination of term rewriting is undecidable, i.e., there is no algorithm that can decide for every finite TRS whether it is terminating.

This can be proved by transforming an arbitrary Turing machine to a TRS and prove that the Turing is halting from every initial configuration if and only the TRS is terminating.

A Turing machine (Q, S, δ) consists of

- a finite set Q of machine states
- a finite set S of tape symbols, including $\square \in S$ representing the blank symbol

- the transition function $\delta : Q \times S \rightarrow Q \times S \times \{L, R\}$

Here $\delta(q, s) = (q', s', L)$ means that if the machine is in state q and reads s , this s is replaced by s' , the machine shifts to the left, and the new machine state is q' .

Similar for $\delta(q, s) = (q', s', R)$: then the machine shifts to the right.

18

This Turing machine behavior can be simulated by a TRS: for a Turing machine $M = (Q, S, \delta)$ we define a TRS $R(M)$ over $Q \cup S \cup \{b\}$ where

- symbols from Q are binary,
- symbols from S are unary and
- b is a constant representing an infinite sequence of blank symbols.

The configuration with tape

$$\dots \square \square \square s'_m s'_{m-1} \dots s'_1 s_1 s_2 \dots s_{n-1} s_n \square \square \square \dots$$

in which the Turing machine is in state q and reads symbol s_1 is represented by the term

$$q(s'_1(s'_2(\dots(s'_m(b))\dots)), s_1(s_2(\dots(s_n(b))\dots)))$$

19

For the Turing machine $M = (Q, S, \delta)$ the TRS $R(M)$ is defined to consist of the rules

$$q(x, s(y)) \rightarrow q'(s'(x), y)$$

for all $(q, s) \in Q \times S$ with $\delta(q, s) = (q', s', R)$, and

$$q(t(x), s(y)) \rightarrow q'(x, t(s'(y)))$$

for all $(q, s) \in Q \times S$ with $\delta(q, s) = (q', s', L)$, for all $t \in S$

and some extra rules representing b to consist of blank symbols.

Theorem

M halts on every configuration if and only if $R(M)$ is terminating.

Consequence: TRS termination is undecidable.

20

Although termination is undecidable, in many special cases termination of a TRS can be proved.

General technique:

Find a weight function W from terms to natural numbers in such a way that $W(u) > W(v)$ for all terms u, v satisfying $u \rightarrow_R v$.

If such a function W exists then R is terminating since an infinite rewriting sequence would give rise to an infinite decreasing sequence of natural numbers which does not exist.

21

In our example

$$\begin{aligned} &+(0, x) \rightarrow x, \\ &+(s(x), y) \rightarrow s(+ (x, y)) \end{aligned}$$

we find such a weight function W by defining inductively

$$\begin{aligned} W(0) &= 1 \\ W(s(t)) &= W(t) + 1 \\ W(+ (t, u)) &= 2W(t) + W(u) \end{aligned}$$

22

The general idea of weight functions is too general:

It allows arbitrary definitions of weight functions, and we have to prove that $W(t) > W(u)$ for **all** rewrite steps $t \rightarrow_R u$, while typically there are infinitely many of them.

Now we work out a special case of this idea of weight functions in such a way that for finding a termination proof we only have to

- choose interpretations for the (finitely many) operation symbols rather than for all terms, and
- check $W(\ell) > W(r)$ for the (finitely many) rules $\ell \rightarrow r$ rather than for all rewrite steps.

23

For every symbol f of arity n choose a **monotonic** function $[f] : \mathbf{N}^n \rightarrow \mathbf{N}$.

Here **monotonic** means:

if for all $a_i, b_i \in \mathbf{N}$ for $i = 1, \dots, n$ with $a_i > b_i$ for some i and $a_j = b_j$ for all $j \neq i$ then

$$[f](a_1, \dots, a_n) > [f](b_1, \dots, b_n)$$

24

Examples:

$$\begin{aligned} &\lambda x \cdot x \\ &\lambda x \cdot x + 1 \\ &\lambda x \cdot 2x \\ &\lambda x, y \cdot x + y \\ &\lambda x, y \cdot x + y + 1 \\ &\lambda x, y \cdot 2x + y \end{aligned}$$

are monotonic

$$\begin{aligned} &\lambda x \cdot 2 \\ &\lambda x, y \cdot x \end{aligned}$$

are **not** monotonic

25

Let \mathcal{X} be the set of variables

For a map $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ the weight function $[\cdot, \alpha] : T \rightarrow \mathbf{N}$ is defined inductively by

$$[x, \alpha] = \alpha(x),$$

$$[f(t_1, \dots, t_n), \alpha] = [f]([t_1, \alpha], \dots, [t_n, \alpha]) \quad \text{and}$$

Theorem

Let R be a TRS and let $[f]$ be chosen such that

- $[f]$ is monotonic for every symbol f , and
- $[\ell, \alpha] > [r, \alpha]$ for every $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ and every rule $\ell \rightarrow r$ in R .

Then R is terminating.

26

Proof sketch:

Assume R admits an infinite reduction

$$t_1 \rightarrow_R t_2 \rightarrow_R t_3 \rightarrow_R \dots$$

Using monotonicity one proves that if $t \rightarrow_R u$ and $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ then $[t, \alpha] > [u, \alpha]$.

Choose $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ arbitrary, then we have

$$[t_1, \alpha] > [t_2, \alpha] > [t_3, \alpha] > \dots$$

being an infinite decreasing sequence of natural numbers, contradiction.

(end of proof sketch)

Often the interpretations $[f]$ are polynomials, therefore the full interpretation is called a **polynomial interpretation**.

27

Example:

For our TRS R consisting of the rules

$$+(0, x) \rightarrow x, \quad +(s(x), y) \rightarrow s(+(x, y))$$

we choose monotonic functions

$$[0] = 1, \quad [s](x) = x + 1,$$

$$[+](x, y) = 2x + y$$

Now indeed for every $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ we have

$$[+(0, x), \alpha] = 2 + \alpha(x) > \alpha(x) = [x, \alpha]$$

$$[+(s(x), y), \alpha] = 2(\alpha(x) + 1) + \alpha(y) >$$

$$(2\alpha(x) + \alpha(y)) + 1 = [s(+(x, y)), \alpha]$$

proving termination.

28

Example:

For the TRS R consisting of the single rule

$$f(g(x)) \rightarrow g(g(f(x)))$$

we choose monotonic functions

$$[f](x) = 3x, \quad [g](x) = x + 1$$

Now indeed for every $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ we have

$$[f(g(x)), \alpha] = 3(\alpha(x) + 1) >$$

$$3\alpha(x) + 1 + 1 = [g(g(f(x))), \alpha]$$

proving termination.

29

Example:

The single rule $f(x) \rightarrow g(f(x))$ is not terminating, but by choosing

$$[f](x) = x + 1, \quad [g](x) = 0$$

for every $\alpha : \mathcal{X} \rightarrow \mathbf{N}$ we have

$$[f(x), \alpha] = \alpha(x) + 1 > 0 = [g(f(x)), \alpha]$$

Where is the error?

$[g]$ is not monotonic!

So monotonicity is essential.

30

Another technique: **lexicographic path order**.

Choose an order $>$ on the set of function symbols.

Theorem

If $\ell >_{lpo} r$ for all $\ell \rightarrow r$ in R , then R is terminating.

Before this makes sense we have to define / characterize $>_{lpo}$

$$f(t_1, \dots, t_n) >_{lpo} u \iff$$

- $\exists i : t_i = u \vee t_i >_{lpo} u$, or
- $u = g(u_1, \dots, u_m)$ and
 - $\forall i : f(t_1, \dots, t_n) >_{lpo} u_i$ and either
 - $f > g$, or
 - $f = g$ and
 - $(t_1, \dots, t_n) >_{lpo}^{lex} (u_1, \dots, u_m)$

31

Lemma:

If s is a proper subterm of t , then $t >_{lpo} s$.

Proof: Easily follows from first bullet.

Example:

For the rule

$$+(0, x) \rightarrow x$$

we have $+(0, x) >_{lpo} x$ by this lemma.

For the rule

$$+(s(x), y) \rightarrow s(+ (x, y))$$

we choose $+ > s$, then by the second item it remains to prove

$$+(s(x), y) >_{lpo} + (x, y)$$

Again using the second item we have to prove

- $+(s(x), y) >_{lpo} x$, which follows from the lemma,
- $+(s(x), y) >_{lpo} y$, which follows from the lemma,

- $(s(x), y) >_{lpo}^{lex} (x, y)$, which follows from $s(x) >_{lpo} x$, which follows again from the lemma.

32

Hence $\ell >_{lpo} r$ for all rules $\ell \rightarrow r$, proving termination.

Checking termination by lexicographic path order is easy to implement; do not choose $>$ in advance but collect requirements on $>$ during the process of proving $\ell >_{lpo} r$.

Several more techniques for proving termination have been developed.

Several tools have been developed by which termination of a TRS can be proved fully automatically: AProVE, TTT2.

33

Recall:

- R is **confluent** (= Church-Rosser, CR):

if $t \rightarrow_R^* u$ and $t \rightarrow_R^* v$ then a term w exists satisfying $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$.

- R is **locally confluent** (= weak Church-Rosser, WCR):

if $t \rightarrow_R u$ and $t \rightarrow_R v$ then a term w exists satisfying $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$.

34

Confluence is strictly stronger than local confluence:

$$\begin{aligned} a &\rightarrow b \\ b &\rightarrow a \\ a &\rightarrow c \end{aligned}$$

$$b \rightarrow d$$

is locally confluent:

if $t \rightarrow_R u$ and $t \rightarrow_R v$ then either

- $t = a$, then choose $w = c$, or
- $t = b$, then choose $w = d$.

In both cases we conclude $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$.

But it is not confluent:

for $t = a, u = c, v = d$ we have $t \rightarrow_R^* u$ and $t \rightarrow_R^* v$, but no w exists satisfying $u \rightarrow_R^* w$ and $v \rightarrow_R^* w$.

35

Newman's lemma (1942):

For terminating TRSs the properties confluence and local confluence are equivalent.

For the proof of Newman's lemma we will use the principle of well-founded induction.

Note that $\text{SN}(\rightarrow)$, $\text{CR}(\rightarrow)$ and $\text{WCR}(\rightarrow)$ all can be defined for arbitrary binary relations \rightarrow , in which general setting we will prove Newman's lemma.

So $\text{SN}(\rightarrow)$ simply means the non-existence of an infinite sequence $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow \dots$.

We write \rightarrow^+ for the transitive closure of \rightarrow : one or more steps.

36

Principle of well-founded induction

If $\text{SN}(\rightarrow)$ and $\forall t (\underbrace{\forall u (t \rightarrow^+ u \Rightarrow P(u))}_{\text{Induction Hypothesis}} \Rightarrow P(t))$

Induction Hypothesis

Then $P(t)$ holds for all t .

(Think of $t \rightarrow^+ u$ as $t > u$, then this coincides with well-known induction.)

Proof of this principle:

Assume there exists t such that $\neg P(t)$.

Then the induction hypothesis does not hold for this t , so $\neg \forall u (t \rightarrow^+ u \Rightarrow P(u))$, yielding u such that $t \rightarrow^+ u$ and $\neg P(u)$.

Repeat the argument for u , yielding a v , and so on, so yielding an infinite sequence

$$t \rightarrow^+ u \rightarrow^+ v \rightarrow^+ \dots$$

contradicting $\text{SN}(\rightarrow)$ (End of proof)

37

Proof of Newman's Lemma

Assume $\text{SN}(\rightarrow)$ and $\text{WCR}(\rightarrow)$, we have to prove $\text{CR}(\rightarrow)$.

So assume $t \rightarrow^* u$ and $t \rightarrow^* v$; we have to find w such that $u \rightarrow^* w$ and $v \rightarrow^* w$.

We apply the principle of well-founded induction for $P(t)$ defined by

$$P(t) \equiv$$

$$\forall u, v : (u \leftarrow^* t \rightarrow^* v \Rightarrow \exists w : u \rightarrow^* w \leftarrow^* v)$$

If $t = u$ we may choose $w = v$.

if $t = v$ we may choose $w = u$.

In the remaining case we have $t \rightarrow^+ u$ and $t \rightarrow^+ v$.

Write $t \rightarrow u_1 \rightarrow^* u$ and $t \rightarrow v_1 \rightarrow^* v$.

38

Using WCR there exists w_1 such that $u_1 \rightarrow^* w_1$ and $v_1 \rightarrow^* w_1$.

Using the induction hypothesis on u_1 there exists w_2 such that $w_1 \rightarrow^* w_2$ and $u \rightarrow^* w_2$.

Now we have $v_1 \rightarrow^* w_2$ and $v_1 \rightarrow^* v$; using the induction hypothesis on v_1 there exists w such that $w_2 \rightarrow^* w$ and $v \rightarrow^* w$.

$$\begin{array}{ccccc}
t & \rightarrow & u_1 & \rightarrow^* & u \\
\downarrow & \text{WCR} & \downarrow^* & \text{IH} & \downarrow^* \\
v_1 & \rightarrow^* & w_1 & \rightarrow^* & w_2 \\
\downarrow^* & & \text{IH} & & \downarrow^* \\
v & & \rightarrow^* & & w
\end{array}$$

Since $u \rightarrow^* w_2$ we have $u \rightarrow^* w$, and we are done.

(End of proof)

39

Both confluence and local confluence are undecidable properties.

However, for terminating TRSs there is a simple decision procedure for local confluence, and hence for confluence too.

Idea:

Analyze overlapping patterns in left hand sides of the rules, yielding **critical pairs**.

In our example there is no overlap, hence our example is locally confluent.

Since we observed it is terminating, by Newman's lemma it is confluent.

40

Definition of critical pairs.

Let $\ell_1 \rightarrow r_1$ and $\ell_2 \rightarrow r_2$ be two (possibly equal) rewrite rules.

Rename variables such that ℓ_1, ℓ_2 have no variables in common.

Let t be a subterm of ℓ_2 , possibly equal to ℓ_2 ; t is not a variable.

Assume t, ℓ_1 **unify**, i.e., there is σ such that $t\sigma = \ell_1\sigma$.

Now $\ell_2\sigma$ can be rewritten in two ways:

- to $r_2\sigma$, and

- to a term u obtained by replacing its subterm $t\sigma = \ell_1\sigma$ by $r_1\sigma$.

In the above situation the pair $[u, r_2\sigma]$ is called a **critical pair**.

The substitution σ can be found in a systematic way if it exists; the result is called **most general unifier (mgu)**.

41

Example:

Assume we have rules for arithmetic including

$$\begin{array}{lcl}
-(x, x) & \rightarrow & 0 \\
-(s(x), y) & \rightarrow & s(-(x, y))
\end{array}$$

Then $-(s(x), s(x))$ can be rewritten in two ways:

- to 0 by the first rule,
- to $s(-(x, s(x)))$ by the second rule.

Now $[0, s(-(x, s(x)))]$ is a **critical pair**.

More precisely, in the above notation we choose

- $\ell_1 \rightarrow r_1$ to be the rule $-(z, z) \rightarrow 0$,
- $\ell_2 \rightarrow r_2$ to be the rule $-(s(x), y) \rightarrow s(-(x, y))$, and
- $t = \ell_2 = -(s(x), y)$.

Indeed t, ℓ_1 unify, with mgu σ :

$$\sigma(x) = x, \quad \sigma(y) = \sigma(z) = s(x)$$

42

Example:

Let R consist of the single rule

$$f(f(x)) \rightarrow g(x)$$

By choosing

- $\ell_1 \rightarrow r_1$ to be the rule $f(f(x)) \rightarrow g(x)$,

- $\ell_2 \rightarrow r_2$ to be the rule $f(f(y)) \rightarrow g(y)$, and
- $t = f(y)$

we see that t, ℓ_1 unify, with mgu σ :

$$\sigma(x) = x, \quad \sigma(y) = f(x)$$

yielding the critical pair $[f(g(x)), g(f(x))]$.

A critical pair $[t, u]$ is said to **converge** if there is a term v such that $t \rightarrow_R^* v$ and $u \rightarrow_R^* v$.

43

Theorem

A TRS R is locally confluent if and only if all critical pairs converge.

Example:

The single rewrite rule $f(f(x)) \rightarrow g(x)$ is not locally confluent, so neither confluent, since for its critical pair $[f(g(x)), g(f(x))]$ no term v exists such that

$$f(g(x)) \rightarrow_R^* v \quad \text{and} \quad g(f(x)) \rightarrow_R^* v$$

This is immediate from the observation that both $f(g(x))$ and $g(f(x))$ are normal forms.

44

For a term t and a TRS R define

$$S(t) = \{v \mid t \rightarrow_R^* v\}$$

If R is finite and terminating then $S(t)$ is finite and computable.

Using the theorem, for a finite terminating TRS R indeed we have an algorithm to decide whether $\text{WCR}(R)$ holds:

- Compute all critical pairs $[t, u]$.

They are found by unification of left hand sides with subterms of left hand sides: there are finitely many of them

- For all critical pairs $[t, u]$ compute

$$S(t) \cap S(u)$$

- If one of these sets is empty then $\text{WCR}(R)$ does not hold.
- If all of these sets are non-empty then $\text{WCR}(R)$ holds.

45

A TRS is said to have **no overlap** if there are only **trivial** critical pairs, i.e., the critical pairs obtained by unifying a left hand side with itself.

A trivial critical pair always converges since it is of the shape $[t, t]$.

As a consequence, every TRS having no overlap is locally confluent.

46

It is not the case that every TRS having no overlap is confluent:

$$\begin{aligned} d(x, x) &\rightarrow b \\ c(x) &\rightarrow d(x, c(x)) \\ a &\rightarrow c(a) \end{aligned}$$

has no overlap but is not confluent:

$$c(a) \rightarrow_R d(a, c(a)) \rightarrow_R d(c(a), c(a)) \rightarrow_R b$$

$$c(a) \rightarrow_R c(c(a)) \rightarrow_R^+ c(b)$$

while $[b, c(b)]$ does not converge.

47

Write \leftrightarrow_R^* for the reflexive symmetric transitive closure of \rightarrow_R , i.e., $t \leftrightarrow_R^* u$ holds if and only if terms t_1, \dots, t_n exist for $n \geq 1$ such that

- $t_1 = t$,
- $t_n = u$, and

- for every $i = 1, \dots, n - 1$ either $t_i \rightarrow_R t_{i+1}$ or $t_{i+1} \rightarrow_R t_i$ holds.

A general question is: given R, t, u , does $t \leftrightarrow_R^* u$ hold?

This is called the **word problem**.

In general the word problem is undecidable.

However, in case R is terminating and confluent the word problem is decidable and admits a simple algorithm.

48

A terminating and confluent TRS is called **complete**.

Now we give a decision procedure for the word problem for complete TRSs.

Rewriting a term t in a terminating TRS as long as possible will always end in a normal form; the result is called a **normal form** of t .

Theorem

If R is a complete TRS and t', u' are normal forms of t, u , then $t \leftrightarrow_R^* u$ if and only if $t' = u'$.

49

For the proof we need a lemma that is easily proved by induction on the length of the path corresponding to $t \leftrightarrow_R^* u$:

Lemma:

If R is confluent and $t \leftrightarrow_R^* u$ then a term v exists such that $t \rightarrow_R^* v$ and $u \rightarrow_R^* v$.

Proof of the theorem:

If $t' = u'$ then $t \rightarrow_R^* t' = u' \leftarrow_R^* u$, hence $t \leftrightarrow_R^* u$.

Conversely assume $t \leftrightarrow_R^* u$.

Then $t' \leftarrow_R^* t \leftrightarrow_R^* u \rightarrow_R^* u'$, hence $t' \leftrightarrow_R^* u'$.

According to the lemma a term v exists such that $t' \rightarrow_R^* v$ and $u' \rightarrow_R^* v$.

Since t', u' are normal forms we have $t' = v = u'$.

(End of proof)

50

The relation \leftrightarrow_R^* is an equivalence relation, and in a complete TRS the normal form is a unique representation for the corresponding equivalence class.

According to the theorem there is a very simple decision procedure for the word problem for complete TRSs:

In order to decide whether $t \leftrightarrow_R^* u$, rewrite

- t to a normal form t' , and
- u to a normal form u' .

Then $t \leftrightarrow_R^* u$ if and only if $t' = u'$.

51

Example:

R consists of the rule $s(s(s(x))) \rightarrow x$.

Does $s^{17}(0) \leftrightarrow_R^* s^{10}(0)$ hold?

We can establish fully automatically that this is not the case:

- Check that R is terminating.
- Check that R is locally confluent.
- Compute the normal form $s(s(0))$ of $s^{17}(0)$.
- Compute the normal form $s(0)$ of $s^{10}(0)$.
- These are different, hence the answer is **No**.

Often a TRS R is not complete, but a complete TRS R' satisfying

$$\leftrightarrow_{R'}^* = \leftrightarrow_R^*$$

can be found in a systematic way.

Finding such a complete TRS is called

(Knuth-Bendix) completion

The new complete TRS can be used for the word problem and unique representation of the original TRS.

Often the original TRS is only a set of equations.

Idea of Knuth-Bendix completion:

Fix a well-founded order $>$ on terms, i.e., $\text{SN}(>)$, that has some closedness properties:

- if $t > u$ then $t\sigma > u\sigma$ for every substitution σ , and
- if $t > u$ then $f(\dots, t, \dots) > f(\dots, u, \dots)$ for every symbol f and every position for t .

Such an order is called a **reduction order**, and has the property:

If $\ell > r$ for every rule $\ell \rightarrow r$ in R , then $\text{SN}(R)$.

A typical example of a reduction order is a lexicographic path order.

Starting with a set E of equations and an empty set R of rewrite rules, repeat the following until E is empty:

Remove an equation $t = u$ from E , and

- add $t \rightarrow u$ to R if $t > u$,
- add $u \rightarrow t$ to R if $u > t$,
- give up otherwise.

After adding any new rule $\ell \rightarrow r$ to R compute all critical pairs between this new rule and existing rules of R , or between the new rule and itself.

For every such critical pair $[t, u]$

- R -rewrite t to normal form t' ,
- R -rewrite u to normal form u' , and
- if $t' \neq u'$, then add $t' = u'$ as an equation to the set E .

What can happen in this Knuth-Bendix procedure?

- It fails due to an equation $t = u$ in E for which neither $t > u$ nor $u > t$ holds.
- It fails since the procedure goes on forever: E gets larger and is never empty.
- It ends with E being empty.

In the last case we really have success: then

- R is terminating since it only contains rule $\ell \rightarrow r$ satisfying $\ell > r$.
- R is locally confluent since all critical pairs converge, so R is complete.
- Convertibility \leftrightarrow_R^* of the resulting R is equivalent to convertibility of the original E since in the whole procedure $\leftrightarrow_{R \cup E}^*$ remains invariant.

Example:

Let E consist of the single equation

$$f(f(x)) = g(x)$$

Choose the lexicographic path order defined by $f > g$.

Since

$$f(f(x)) >_{lpo} g(x)$$

we add the rule $f(f(x)) \rightarrow g(x)$ to the empty TRS R .

Now the critical pair $[f(g(x)), g(f(x))]$ gives rise to the new equation $f(g(x)) = g(f(x))$ in E .

Since

$$f(g(x)) >_{lpo} g(f(x))$$

we add the rule $f(g(x)) \rightarrow g(f(x))$ to the TRS R .

Together with the older rule $f(f(x)) \rightarrow g(x)$ we get the critical pair $[f(g(f(x))), g(g(x))]$.

Since $g(g(x))$ is a normal form and

$$f(g(f(x))) \rightarrow_R g(f(f(x))) \rightarrow_R g(g(x))$$

no new equation is added to E , and E is empty.

So we end up in the complete TRS R consisting of the two rules

$$f(f(x)) \rightarrow g(x), \quad f(g(x)) \rightarrow g(f(x))$$

having the same convertibility relation as the original equation $f(f(x)) = g(x)$.