Termination of Term Rewriting

Hans Zantema
Dept. of Computer Science, Universiteit Utrecht,
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands,
and
CWI, P.O. Box 94.079, 1090 GB Amsterdam, The Netherlands,
E-mail: hansz@cs.uu.nl

Contents

1 Introduction

Given a TRS (Σ, R), how to prove or disprove that it is terminating? It is well-known ([35]) that this question is undecidable for finite TRSs. This means that it is impossible to develop a method that decides whether a given finite TRS is terminating or not. However, we may expect that there are methods that can be used to establish termination for many TRSs occurring in practical situations. Indeed such methods have been developed, and in this chapter an overview of them is given.

Roughly speaking three kinds of methods can be distinguished: *semantical methods*, *syntactical methods* and *transformational methods*. Proving termination by a semantical method means that a weight function has to be defined in such a way that by every reduction step the weight of a term strictly decreases. If the weight is a natural number, or, more generally, a value in a set equipped with a well-founded order, then this cannot go on forever so termination is ensured. As a framework this is the most basic idea for proving termination; it serves as the basis of a hierarchy of different kinds of termination. About how to choose suitable weight functions only some rough heuristics are available. This approach easily generalizes to termination modulo equations. A special case of this approach is the method of *polynomial interpretations*.

Syntactical methods are based upon orders on terms that are defined by induction on the structure of the terms. Given such an order, if it can be proved that it is well-founded, and if every reduction step causes a decrease with respect to the order, then termination has been proved. Taking the set of terms equipped with such an order makes this approach fit in the general framework of semantical methods. A typical order of this kind is the *recursive path order*. Orders of this kind are usually *precedence based*: they depend upon an order (called *precedence*) on the function symbols. The advantage of this method is that it provides an algorithm that checks whether termination can be proved by this method or not. This algorithm is easy to implement. The disadvantage is that it covers only a very restricted class of terminating systems. Generalization to termination modulo equations is hardly possible.

Transformational methods provide non-termination preserving transformations between rewrite systems: a transformation Φ falls in this category if termination of a TRS (Σ, R) follows from termination of $\Phi(\Sigma, R)$. If such a Φ can be found in such a way that termination of $\Phi(\Sigma, R)$ can be proved by any other method, then termination of (Σ, R) has been proved. This approach easily generalizes to termination modulo equations.

On each of these kinds of methods a section will be spent. After the treatment of semantical methods a hierarchy of kinds of termination will be given, based on different kinds of well-founded orders in which the weights are taken.

For most methods the notion of order plays an important role. Before the various methods are discussed some basic facts due to Lankford ([46]) about the connection between termination of TRSs and orders are treated. A strict order $<$ is called *well-founded* if it does not admit an infinite descending sequence $t_0 > t_1 > t_2 > t_3 \cdots$.

Proposition 1 *A TRS (Σ, R) is terminating if and only if there exists a well-founded order $<$ on $Ter(\Sigma)$ such that $t > u$ for every $t, u \in Ter(\Sigma)$ for which $t \to_R u$.*

Proof: If (Σ, R) is terminating then choose $<$ to be $\leftarrow_R^+$. On the other hand if $<$ is an order satisfying the requirements then termination of (Σ, R) follows from well-foundedness of $<$. $\Box$

Since usually there are infinitely many pairs of terms $t, u \in Ter(\Sigma)$ for which $t \to_R u$, this proposition is only rarely useful for proving termination of a TRS. In the next proposition we see that with an extra condition on the order it suffices to check $t > u$ only for the rewrite rules instead of for all rewrite steps. Since usually the number of rewrite rules is finite this is much more convenient. First we need a definition.

Definition 2 *A* reduction order *on $Ter(\Sigma)$ is a well-founded order $<$ on $Ter(\Sigma)$ that is*

- *closed under substitutions, i.e., if $t < u$ and σ is an arbitrary substitution then $t^\sigma < u^\sigma$, and*
- *closed under contexts, i.e., if $t < u$ and C is an arbitrary context then $C[t] < C[u]$.*

A reduction order $<$ on $Ter(\Sigma)$ is called compatible *with a TRS (Σ, R) if $l > r$ for every rewrite rule $l \to r$ in R.*

Proposition 3 *A TRS (Σ, R) is terminating if and only if it admits a compatible reduction order $<$ on $Ter(\Sigma)$.*

Proof: If (Σ, R) is terminating then again choose $<$ to be $\leftarrow_R^+$. By definition $<$ is closed under substitutions and contexts.

On the other hand, if $<$ is a reduction order that is compatible with (Σ, R), then for an arbitrary rewrite step $t \equiv C[l^\sigma] \to_R C[r^\sigma] \equiv u$ we obtain $t > u$ due to compatibility and closedness under substitutions and contexts. Now termination of (Σ, R) follows from well-foundedness of $<$. □

2 Semantical methods

Consider the following two simple examples of TRSs.

$$R_1 \left\{ \begin{array}{lcl} f(f(x)) & \to & g(x) \\ g(g(x)) & \to & f(x) \end{array} \right. \qquad R_2 \left\{ \begin{array}{lcl} f(g(x)) & \to & f(h(f(x))) \\ g(g(x)) & \to & f(g(h(x))) \end{array} \right.$$

Many people immediately see why these TRSs are terminating: R_1 is terminating since at each reduction step the size of the term strictly decreases, and R_2 is terminating since at each reduction step the number of g-symbols strictly decreases. For checking that indeed this kind of weight decreases at every reduction step, it suffices to check that for every rule the weight of the left hand side is strictly greater than the weight of the right hand side. In this section this kind of termination proofs is formalized and generalized.

2.1 Well-founded monotone algebras

Let Σ be a signature, being a set of operation symbols each having a fixed arity.[1] A Σ-algebra (A, Σ_A) is defined to consist of a non-empty set A, and for every $f \in \Sigma$ a function $f_A : A^n \to A$, where n is the arity of f. This function f_A is called the *interpretation* of f; we write $\Sigma_A = \{f_A \mid f \in \Sigma\}$.

[1] Having a fixed arity is not an essential restriction: if a symbol allows more arities, simply replace it by distinct symbols, one for every allowed arity.

Definition 4 *A* well-founded monotone Σ-algebra $(A, \Sigma_A, <)$ *is a* Σ*-algebra* (A, Σ_A) *equipped with a well-founded order* $<$ *on* A *such that each algebra operation is strictly monotone in every argument. More precisely, for every* $f \in \Sigma$ *and all* $a_1, \ldots, a_n, b_1, \ldots, b_n \in A$ *with* $a_i < b_i$ *for some* i *and* $a_j = b_j$ *for all* $j \neq i$ *we have*

$$f_A(a_1, \ldots, a_n) < f_A(b_1, \ldots, b_n).$$

Let $(A, \Sigma_A, <)$ be a well-founded monotone Σ-algebra. As before $Ter(\Sigma)$ denotes the set of terms over Σ and a set $\mathcal{X}$ of variable symbols. Let $\alpha : \mathcal{X} \to A$. We define the term evaluation

$$[\![-]\!]_\alpha : Ter(\Sigma) \to A$$

inductively by

$$\begin{aligned} [\![x]\!]_\alpha &= \alpha(x), \\ [\![f(t_1, \ldots, t_n)]\!]_\alpha &= f_A([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha) \end{aligned}$$

for $x \in \mathcal{X}, f \in \Sigma, t_1, \ldots, t_n \in Ter(\Sigma)$. This function induces a strict partial order $<_A$ on $Ter(\Sigma)$ as follows:

$$t <_A t' \;\Leftrightarrow\; \forall \alpha : \mathcal{X} \to A \;:\; [\![t]\!]_\alpha < [\![t']\!]_\alpha.$$

Stated in words, $t <_A t'$ means that for each interpretation of the variables in A the interpreted value of t is smaller than that of t'.

We say that a well-founded monotone algebra $(A, \Sigma_A, <)$ is *compatible* with a TRS if $l >_A r$ for every rule $l \to r$ in the TRS.

Now we arrive at the basic result for this section, which essentially goes back to an unpublished note of Lanford of 1975.

Theorem 5 *A TRS is terminating if and only if it admits a compatible well-founded monotone algebra.*

In order to give the proof we need two lemmas.

Lemma 6 *Let* $\sigma : \mathcal{X} \to Ter(\Sigma)$ *be any substitution and let* $\alpha : \mathcal{X} \to A$. *Let* $\beta : \mathcal{X} \to A$ *be defined by* $\beta(x) = [\![\sigma(x)]\!]_\alpha$. *Then*

$$[\![t^\sigma]\!]_\alpha = [\![t]\!]_\beta$$

for all $t \in Ter(\Sigma)$.

Proof: Induction on the structure of t. □

Lemma 7 *For any well-founded monotone algebra* $(A, \Sigma_A, <)$ *the partial order* $<_A$ *is a reduction order.*

Proof: Well-foundedness of $<_A$ is immediate from well-foundedness of $<$. It remains to show that $<_A$ is closed under substitutions and contexts. Let $t <_A t'$ for $t, t' \in Ter(\Sigma)$ and let $\sigma : \mathcal{X} \to Ter(\Sigma)$ be any substitution. Let $\alpha : \mathcal{X} \to A$. From Lemma 6 we obtain

$$[\![t^\sigma]\!]_\alpha = [\![t]\!]_\beta < [\![t']\!]_\beta = [\![t'^\sigma]\!]_\alpha$$

for some β depending on σ and α. This holds for all $\alpha : \mathcal{X} \to A$, so $t^\sigma <_A t'^\sigma$. Hence $<_A$ is closed under substitutions.

For proving closedness under contexts assume $t <_A t'$ for $t, t' \in Ter(\Sigma)$, and let $f \in \Sigma$. Since $t <_A t'$ we have $[\![t]\!]_\alpha < [\![t']\!]_\alpha$ for all $\alpha : \mathcal{X} \to A$. Applying the monotonicity condition of f_A we obtain

$$\begin{aligned} [\![f(\ldots,t,\ldots)]\!]_\alpha &= f_A(\ldots,[\![t]\!]_\alpha,\ldots) \\ &< f_A(\ldots,[\![t']\!]_\alpha,\ldots) \\ &= [\![f(\ldots,t',\ldots)]\!]_\alpha. \end{aligned}$$

This holds for all $\alpha : \mathcal{X} \to A$, so

$$f(\ldots,t,\ldots) <_A f(\ldots,t',\ldots).$$

Closedness under arbitrary contexts follows from this result by induction on the context. □

Now we give the proof of Theorem 5.

Proof: If a TRS admits a compatible well-founded monotone algebra $(A, \Sigma_A, <)$ then termination follows from Lemma 7 and Proposition 3.

On the other hand, assume the system is terminating. Define $A = Ter(\Sigma)$, let $f_A(t_1,\ldots,t_n) = f(t_1,\ldots,t_n)$, and define $<$ to be the transitive closure of the inverse of the rewrite relation. One easily verifies that $(A, \Sigma_A, <)$ is a well-founded monotone algebra. We still have to prove that $l >_A r$ for each rewrite rule $l \to r$. Let $\alpha : \mathcal{X} \to A$. Since $A = Ter(\Sigma)$ we see that α is a substitution. Then $[\![t]\!]_\alpha = t^\alpha$ for each term t, which is easily proved by induction on the structure of t. Since $l \to r$ is a rewrite rule, the term l^α can be reduced in one step to r^α. So

$$[\![l]\!]_\alpha = l^\alpha > r^\alpha = [\![r]\!]_\alpha.$$

This holds for every $\alpha : \mathcal{X} \to A$, so $l >_A r$. □

A way of proving termination of a TRS is now as follows: choose a set A equipped with a well-founded order $<$, define for each operation symbol f a corresponding operation f_A that is strictly monotone in all of its arguments, and for which $[\![l]\!]_\alpha > [\![r]\!]_\alpha$ for all rewrite rules $l \to r$ and all $\alpha : \mathcal{X} \to A$. Then according to Theorem 5 the TRS is terminating.

The problem is how to choose the partially ordered set and the operations. The simplest useful choice for $(A, <)$ is $(\mathbb{N}^+, <)$, the set of strictly positive integers with the ordinary order. In many applications this is already a fruitful choice.

Example 1 The two simple examples in the beginning of this section can be seen as applications of Theorem 5 with this choice for $(A, <)$ as follows. The size of terms over f, g is modelled by choosing $f_A(x) = g_A(x) = x + 1$ for all $x \in A$. Indeed f_A and g_A are both strictly monotone, and

$$f_A(f_A(x)) = x + 2 > x + 1 = g_A(x) \quad \text{and} \quad g_A(g_A(x)) = x + 2 > x + 1 = f_A(x)$$

for all $x \in A$. Hence $f(f(x)) >_A g(x)$ and $g(g(x)) >_A f(x)$, proving termination of R_1 by Theorem 5. The number of g's in terms over f, g, h is modelled by choosing $f_A(x) = h_A(x) = x$ and $g_A(x) = x + 1$ for all $x \in A$. Clearly f_A, g_A and h_A are all strictly monotone, and

$$f_A(g_A(x)) = x + 1 > x = f_A(h_A(f_A(x))) \quad \text{and}$$

$$g_A(g_A(x)) = x + 2 > x + 1 = f_A(g_A(h_A(x)))$$

for all $x \in A$, proving termination of R_2 by Theorem 5.

Example 2 As another application of the choice $(A, <) = (\mathbb{N}^+, <)$ we consider associativity. Consider the system consisting of one single rule

$$f(f(x,y),z) \ \rightarrow \ f(x,f(y,z))$$

and choose $f_A(x,y) = 2x + y$ for all $x, y \in A$. Clearly f_A is strictly monotone in both arguments, and

$$f_A(f_A(x,y),z) = 4x + 2y + z > 2x + 2y + z = f_A(x, f_A(y,z))$$

for all $x, y, z \in A$, proving termination according to Theorem 5.

Exercise 1 Prove termination of the following TRSs, each consisting of a single rule, by choosing $(A, <) = (\mathbb{N}^+, <)$.

- $f(g(x)) \rightarrow g(f(x))$;
- $f(g(x)) \rightarrow g(g(f(x)))$;
- $f(g(x)) \rightarrow a(g(f(x)), x)$.

Theorem 5 admits many generalizations. In paricular, it extends to rewriting modulo equations and to context-sensitive term rewriting. Context-sensitive term rewriting is a kind of term rewriting in which reduction is not allowed inside some fixed arguments of some function symbols. By restricting the monotonicity requirements for f_A to the arguments of f in which reduction is allowed, a generalized notion of monotone algebras is obtained for which a context-sensitive TRS is terminating if and only if admits a compatible generalized monotone algebra; for details we refer to [67].

We conclude this subsection by extending Theorem 5 to termination modulo equations. A TRS (Σ, R) is called terminating modulo a set E of equations over Σ if no infinite sequence of the following shape exists:

$$t_1 \rightarrow_R t_2 =_E t_3 \rightarrow_R t_4 =_E t_5 \rightarrow_R t_6 \ldots .$$

In typical applications of this notion the set E consists of commutativity and associativity for one or more binary symbols, usually abbreviated to AC.

We say that a Σ-algebra (A, Σ_A) *satisfies* a set E of equations over Σ if $[\![t]\!]_\alpha = [\![u]\!]_\alpha$ for all $\alpha : \mathcal{X} \rightarrow A$ and all equations $t = u$ in E.

Theorem 8 *A TRS (Σ, R) is terminating modulo a set E of equations if and only if R admits a compatible well-founded monotone algebra satisfying E.*

Proof: For the 'if'-part assume that $(A, \Sigma_A, <)$ is a compatible well-founded monotone algebra satisfying E and an infinite sequence

$$t_1 \rightarrow_R t_2 =_E t_3 \rightarrow_R t_4 =_E t_5 \rightarrow_R t_6 \ldots .$$

exists. Let $\alpha : \mathcal{X} \rightarrow A$ be arbitrary (here we use the assumption that A is not empty), then compatibility, Lemma 7 and the fact that A satisfies E yield

$$[\![t_1]\!]_\alpha > [\![t_2]\!]_\alpha = [\![t_3]\!]_\alpha > [\![t_4]\!]_\alpha = [\![t_5]\!]_\alpha > [\![t_6]\!]_\alpha \ldots ,$$

contradicting well-foundedness.

For the 'only if'-part assume that R is terminating modulo E. Define $A = Ter(\Sigma)/=_E$, i.e., A consists of the equivalence classes $[t]$ of terms t modulo the

equivalence relation $=_E$. For $f \in \Sigma$ of arity n the operation $f_A : A^n \to A$ is defined by

$$f_A([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)];$$

due to the congruence property of $=_E$ the result is independent of the choice of representatives in the classes $[t_i]$. Now we define

$$[t] <' [u] \quad \Leftrightarrow \quad \exists t', u' \; : \; t =_E t' \wedge u' \to_R t' \wedge u' =_E u.$$

One easily checks that $<'$ is well-defined. We define $<$ to be the transitive closure of $<'$. Since R is terminating modulo E we conclude that $(A, \Sigma_A, <)$ is a well-founded monotone algebra. We still have to verify that $l >_A r$ for each rewrite rule $l \to r$ of R. This is similar to the corresponding part of the proof of Theorem 5. □

2.2 Polynomial interpretations

In the basic applications of Theorem 5 the well-founded monotone algebra $(A, \Sigma_A, <)$ consists of the natural numbers with the usual order, and for all $f \in \Sigma$ the function f_A is a polynomial in its arguments. For this kind of polynomial interpretations some heuristics have been developed to choose the polynomials, and some techniques to prove that $l >_A r$ for rewrite rules $l \to r$ ([11]). Here we give an overview of these techniques. We start with a number of definitions.

Definition 9 *A* monomial *in n variables over $\mathbb{Z}$ is a function $F : \mathbb{Z}^n \to \mathbb{Z}$ defined by $F(x_1, \ldots, x_n) = a x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$ for some integer $a \neq 0$ and some non-negative integers $k_1, k_2, \ldots, k_n$. The number a is called the* coefficient *of the monomial. If $k_1 = k_2 = \cdots = k_n = 0$ then the monomial is called a* constant.

For functions $F_i : \mathbb{Z}^n \to \mathbb{Z}$, $i = 1, \ldots, m$, the sum *and the* product *are defined respectively as follows:*

$$(\sum_{i=1}^{m} F_i)(x_1, \ldots, x_n) = \sum_{i=1}^{m} F_i(x_1, \ldots, x_n),$$

$$(\prod_{i=1}^{m} F_i)(x_1, \ldots, x_n) = \prod_{i=1}^{m} F_i(x_1, \ldots, x_n).$$

A polynomial *in n variables over $\mathbb{Z}$ is the sum of finitely many monomials in n variables over $\mathbb{Z}$.*

Choose

$$A = \{n \in \mathbb{N} \mid n \geq 2\}.$$

A polynomial interpretation *for a signature Σ consists of a polynomial f_A in n variables for every symbol $f \in \Sigma$, where n is the arity of f, such that for all $f \in \Sigma$:*

- *$f_A(x_1, \ldots, x_n) \in A$ for all $x_1, \ldots, x_n \in A$ (*well-definedness*), and*
- *f_A is strictly monotone in all of its arguments.*

Now $(A, \Sigma_A, <)$ is a well-founded monotone algebra and the order $<_A$ is defined as before.

A polynomial interpretation $\{f_A \mid f \in \Sigma\}$ is compatible *with a TRS (Σ, R) if $l >_A r$ for every rule $l \to r$ in R, or stated equivalently, if the well-founded monotone algebra $(A, \Sigma_A, <)$ is compatible with (Σ, R).*

A TRS is polynomially terminating *if it admits a compatible polynomial interpretation.*

Usually the monomial F defined by $F(x_1, \ldots, x_n) = x_i$ is denoted by X_i, for $i = 1, \ldots, n$; if $n \leq 3$ one often writes $X = X_1, Y = X_2, Z = X_3$. A constant monomial is denoted by its coefficient. In this way the monomial $F : \mathbb{Z}^n \to \mathbb{Z}$ defined by $F(x_1, \ldots, x_n) = a x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$ for an integer $a \neq 0$ and non-negative integers $k_1, k_2, \ldots, k_n$ is written as $a X_1^{k_1} X_2^{k_2} \cdots X_n^{k_n}$.

One easily sees that

- the product of finitely many monomials is again a monomial,
- the sum of finitely many polynomials is again a polynomial,
- the product of finitely many polynomials is again a polynomial.

For instance, $3XZ + Y^3$ is a polynomial.

Due to Theorem 5 polynomial termination indeed implies termination. We like to stress here that strict monotonicity in all arguments is essential for this statement. For instance, the rule $f(s(x), y) \to f(x, f(s(x), y))$ is not terminating although $f(s(x), y) >_A f(x, f(s(x), y))$ holds in the interpretation $f_A = X, s_A = X + 1$. This interpretation does not fulfil the requirements of Theorem 5: f_A is not strictly monotone in its second argument.

The reason for choosing $A = \{n \in \mathbb{N} \mid n \geq 2\}$ instead of $A = \mathbb{N}$ is merely for convenience. For the power of the interpretations this does not have any effect since this set is order-isomorphic to $\mathbb{N}$ by an order-isomorphism adding 2 to every element, and the composition of polynomials and such isomorphisms again yield polynomials. But this choice has a few advantages over the natural numbers, for instance excluding zero yields that multiplication is strictly monotone in both arguments, and excluding 1 yields that the polynomial XY pointwise exceeds the polynomials X and Y. As a drawback of this choice for a constant we may not choose a value less than 2.

The following proposition gives sufficient (but not necessary) conditions for well-definedness and strict monotonicity.

Proposition 10 *Let $F = \sum_{i=1}^{m} a_i X_1^{k_{i,1}} X_2^{k_{i,2}} \cdots X_n^{k_{i,n}}$ be a polynomial in $n > 0$ variables for which $a_i > 0$ for $i = 1, \ldots, m$, and for every $j = 1, \ldots, n$ there exists i with $1 \leq i \leq m$ and $k_{i,j} > 0$. Then F is well-defined, and F is strictly monotone in all its arguments.*

Proof: Every non-constant monomial with coefficient ≥ 1 yields a value ≥ 2 for every interpretation of the variables in A. Since $n > 0$ there exists $k_{i,j} > 0$, hence at least one of the monomials is non-constant. Hence F is well-defined.

A monomial $a X_1^{k_1} X_2^{k_2} \cdots X_n^{k_n}$ with $a > 0$ is strictly monotone in the i-th argument if $k_i > 0$, and is weakly monotone in all other arguments. Hence F is strictly monotone in all its arguments. □

The condition in Proposition 10 that every variable occurs in at least one of the monomials can not be weakened: if some variable does not occur in any of the monomials then the polynomial is not strictly monotone in the corresponding argument. For instance, the polynomial $Y + YZ^2$ in 3 variables X, Y, Z is not strictly monotone in the first argument.

The condition in Proposition 10 that all coefficients are positive is not necessary: for instance $X^2 - X$ is a well-defined and strictly monotone polynomial in one variable. However, in most applications negative coefficients do not occur and Proposition 10 can be applied for concluding well-definedness and strict monotonicity.

For a rewrite rule $l \to r$ let $x_1, \ldots, x_n$ be the variables occurring in l and r. Given a polynomial interpretation define $F_{l,r} : A^n \to \mathbb{Z}$ by

$$F_{l,r}(a_1, \ldots, a_n) = [\![l]\!]_\alpha - [\![r]\!]_\alpha$$

for α defined by $\alpha(x_i) = a_i$ for $i = 1, \ldots, n$. One easily checks that $F_{l,r}$ is a polynomial in n variables; often some coefficients will be negative. By definition $l >_A r$ if and only if $F_{l,r}(a_1, \ldots, a_n) > 0$ for all $a_1, \ldots, a_n \in A$. A polynomial F is called *strictly positive* if $F(a_1, \ldots, a_n) > 0$ for all $a_1, \ldots, a_n \in A$. The next proposition states that strict positiveness is undecidable.

Proposition 11 *Given an arbitrary polynomial F over $\mathbb{Z}$ in $n \geq 2$ variables it is undecidable whether $F(a_1, \ldots, a_n) > 0$ for all $a_1, \ldots, a_n \in A$.*

Proof: For an arbitrary polynomial F over $\mathbb{Z}$ it is undecidable whether there exist $a_1, \ldots, a_n \in \mathbb{Z}$ with $F(a_1, \ldots, a_n) = 0$. This was proved by Y. Matiyasevich in 1970 and solved Hilbert's tenth problem ([56]).

Assume that a decision procedure exists deciding whether $F(a_1, \ldots, a_n) > 0$ for all $a_1, \ldots, a_n \in A$ for any F to be given. Since for every $x \in \mathbb{Z}$ we have $x \geq 2$ or $4 - x \geq 2$ we can write $x = f(a)$ for $a \in A$ and f is either the polynomial X or the polynomial $4 - X$. Let F be an arbitrary polynomial over $\mathbb{Z}$ in n variables. Then

$$\exists a_1, \ldots, a_n \in \mathbb{Z} \ : \ F(a_1, \ldots, a_n) = 0 \ \Leftrightarrow$$

$$\neg \forall a_1, \ldots, a_n \in \mathbb{Z} \ : \ F(a_1, \ldots, a_n) \neq 0 \ \Leftrightarrow$$

$$\neg \forall a_1, \ldots, a_n \in \mathbb{Z} \ : \ (F(a_1, \ldots, a_n))^2 > 0 \ \Leftrightarrow$$

$$\neg (\bigwedge_{f_i = X \vee f_i = 4 - X} \forall a_1, \ldots, a_n \in A \ : \ (F(f_1(a_1), \ldots, f_n(a_n)))^2 > 0);$$

in the last line the conjunction runs over all 2^n choices of f_i being either X or $4 - X$ for $i = 1, \ldots, n$. By applying the assumed decision procedure on all of these 2^n conjuncts this yields a decision procedure for Hilbert's tenth problem, contradiction. □

According to [36] Hilbert's tenth problem is even undecidable if only polynomials in $n \leq 9$ variables are considered. Since the number of variables is preserved by the construction in the proof of Proposition 11, it also holds for polynomials in $n \leq 9$ variables. But also for polynomials of a low degree in far less variables, checking for positiveness or for zeroes seems to be practically unfeasible. As an example we mention that the polynomial $F = X^2 - 991Y^2 - 1$ indeed admits integer values $a, b > 1$ with $F(a, b) = 0$, but only with values exceeding 10^{28}.

Surprisingly, the problem of deciding strict positiveness of polynomials over the real numbers instead of over the integers is decidable, by means of techniques from algebraic geometry. In practice these techniques are not useful; for our purpose much simpler techniques suffice for proving strict positiveness of polynomials.

Back to rewriting. In order to prove termination of a TRS (Σ, R) by means of a polynomial interpretation we have to choose a polynomial f_A in n variables for every symbol $f \in \Sigma$, where n is the arity of f, satisfying the following three conditions:

- The polynomial f_A is well-defined. For a constant this means that we have to choose a value ≥ 2; if the arity of f is positive and f_A is of the required shape this is obtained for free by Proposition 10.

- The polynomial f_A is strictly monotone in all of its arguments. For a constant this condition is empty, if the arity of f is positive and f_A is of the required shape this is obtained for free by Proposition 10.
- The interpretation has to be compatible. This means that $F_{l,r}(a_1, \ldots, a_n) > 0$ for all $a_1, \ldots, a_n \in A$ and every rule $l \to r$.

We conclude that the method now consists of two main steps:

- Choose suitable polynomials with positive coefficients for all symbols $f \in \Sigma$ in which all variables occur.
- Prove that $F_{l,r}(a_1, \ldots, a_n) > 0$ for all $a_1, \ldots, a_n \in A$ for the corresponding polynomials $F_{l,r}$ for all rules $l \to r$ of R.

Regarding the first step only some rough heuristics can be given, after giving a number of examples we will summarize a few. Often the first choice is not yet successful and some backtracking is applied for finding the right choice.

For the second step we now describe a technique from [11]. It is based on the simple observation that if $k_{q,j} \geq k_{p,j} \geq 0$ for all $j = 1, \ldots, n$, and $c = 2^{\sum_{j=1}^{n}(k_{q,j} - k_{p,j})}$, then $x_1^{k_{q,1}-k_{p,1}} x_2^{k_{q,2}-k_{p,2}} \cdots x_n^{k_{q,n}-k_{p,n}} \geq c$ for all $x_1, x_2, \ldots, x_n \in A$, and hence

$$x_1^{k_{q,1}} x_2^{k_{q,2}} \cdots x_n^{k_{q,n}} - c x_1^{k_{p,1}} x_2^{k_{p,2}} \cdots x_n^{k_{p,n}} \geq 0$$

for all $x_1, x_2, \ldots, x_n \in A$.

Let $F = \sum_{i=1}^{m} a_i X_1^{k_{i,1}} X_2^{k_{i,2}} \cdots X_n^{k_{i,n}}$ be a polynomial of which strict positiveness has to be proved. This is done by the next non-deterministic procedure, trying to stepwise decrease the polynomial until a strictly positive polynomial is obtained only having positive coefficients.

```
while ∃p : a_p < 0
  do
  choose p, q satisfying a_p < 0 ∧ a_q > 0 ∧ ∀j ∈ {1,...,n} : k_{q,j} ≥ k_{p,j};
  c := 2^{Σ_{j=1}^{n} (k_{q,j} − k_{p,j})};
  if a_p + c a_q > 0 then a_q := a_q + (a_p / c);
                          a_p := 0
                     else a_p := a_p + c a_q;
                          a_q := 0
  fi
  od
```

If this procedure ends in a situation in which no p exists with $a_p < 0$ and at least one remaining coefficient is strictly positive, then the last version of the polynomial is strictly positive. Since during the whole procedure the polynomial only decreases, this proves that also the original polynomial is strictly positive.

For example, assume that we have to prove that $l >_A r$ where $[\![l]\!]_\alpha = b_1 b_2^2 + b_1^2$ and $[\![r]\!]_\alpha = 3b_2 + 5$ for α defined by $\alpha(x_i) = b_i$ for $i = 1, 2$. Then we have to prove strict positiveness of the polynomial $XY^2 + X^2 - 3Y - 5$. Let the four coefficients 1, 1, -3 and -5 be a_1, a_2, a_3, a_4, respectively. The above procedure may choose $p = 3$ and $q = 1$ for its first step, trying to eliminate $-3Y$ by decreasing the coefficient of XY^2. This succeeds by replacing the polynomial $XY^2 + X^2 - 3Y - 5$ by

$$\frac{1}{4} XY^2 + X^2 - 5.$$

In the next step the above procedure may choose $p = 4$ and $q = 2$, trying to eliminate -5 by decreasing the coefficient of X^2. For this the coefficient of X^2 is too small, and X^2 is eliminated, yielding

$$\frac{1}{4}XY^2 - 1.$$

Finally, in order to eliminate -1 the above procedure chooses $p = 4$ and $q = 1$, yielding

$$\frac{1}{8}XY^2.$$

Now indeed the only remaining coefficient is positive, hence this last polynomial is strictly positive. Hence the same holds for the original polynomial, successfully finishing the proof. Note that in general during the process the coefficients do not remain integer; powers of 2 appear as denominators.

If during the procedure at least one value of p with $a_p < 0$ remains, but for none of them a value q exists satisfying $a_q > 0 \wedge \forall j = 1, \ldots, n \ : \ k_{q,j} \geq k_{p,j}$, then the procedure fails to prove strict positiveness of the original polynomial. The same holds if the procedure ends in a situation in which $a_p = 0$ for all $p = 1, \ldots, n$.

Often the procedure allows more than one possibility for choosing suitable p and q. Sometimes the success of the procedure depends on this choice. For instance, a failing execution of the procedure of the same example as above is obtained by choosing $p = 4$ and $q = 1$ in the first step and $p = 3$ and $q = 1$ in the second step, yielding

$$XY^2 + X^2 - 3Y - 5$$
$$\frac{3}{8}XY^2 + X^2 - 3Y$$
$$X^2 - \frac{3}{2}Y.$$

A good heuristics for the procedure is choosing p, q in such a way that $\sum_{j=1}^{n}(k_{q,j} - k_{p,j})$ is as small as possible.

Exercise 2 Prove strict positiveness of the following polynomials over the natural numbers ≥ 2:

$$X^4 - X^3 + X^2 - 11,$$
$$XY - X - Y + 1,$$
$$XYZ - X - Y - Z - 1,$$
$$X^2Y + Y^2 - X - 3Y.$$

The procedure allows lots of modifications and improvements. The basic idea of the procedure is that for proving strict positiveness of a polynomial it suffices to prove strict positiveness after subsequently subtracting some polynomials known to be non-negative. In the version presented here only the fact is used that the product of k values from A is always $\geq 2^k$ by choosing the required non-negative polynomials all being positive multiples of polynomials of the shape

$$X_1^{k_{q,1}} X_2^{k_{q,2}} \cdots X_n^{k_{q,n}} - cX_1^{k_{p,1}} X_2^{k_{p,2}} \cdots X_n^{k_{p,n}}.$$

The present version is not able to establish strict positiveness of $X^2 + Y^2 - XY$, for example. The method can be improved covering this example by making use of more kinds of always non-negative polynomials, like $(X - 2)^2$ and $(X - Y)^2$. However, in practice the version discussed here suffices very well, and due to Proposition 11 no improvement of the procedure will cover all strictly positive polynomials.

There is no fundamental reason to restrict to polynomials. In [47] this approach has been extended to so-called *elementary functions*: functions for which the building blocks addition and multiplication for polynomials have been extended by exponentiation.

We conclude this section by some examples and some heuristics for choosing the polynomials.

Example 3

$$\begin{array}{rcl} 0+x & \to & x \\ s(x)+y & \to & s(x+y) \\ 0*x & \to & 0 \\ s(x)*y & \to & y+(x*y) \end{array}$$

This system describes the usual arithmetic on natural numbers. Choose $0_A = 2, s_A = X+3, +_A = 2X+Y, *_A = XY$. Then the polynomials $F_{l,r}$ for the four rules are $4, 3, 2X-2, Y$, respectively, all of which are clearly strictly positive, proving polynomial termination.

Example 4

$$\begin{array}{rcl} x+x & \to & x \\ (x+y)\cdot z & \to & (x\cdot z)+(y\cdot z) \\ (x\cdot y)\cdot z & \to & x\cdot(y\cdot z) \\ x+\delta & \to & x \\ \delta\cdot x & \to & \delta \end{array}$$

This system describes part of a process algebra where $+$ denotes choice, $\cdot$ denotes sequential composition and δ denotes deadlock. Choose $\delta_A = 2, +_A = X+Y, \cdot_A = X^2Y$. Now the polynomials $F_{l,r}$ for the five rules are $X, 2XYZ, X^4Y^2Z - X^2Y^2Z, 2, 4X-2$, respectively, all of which are checked to be strictly positive automatically by the above method, proving polynomial termination.

By combining Propositions 21 and 29 from Section 3.1, we will see that rules for which the right hand side can be embedded in the left hand side, don't need to be considered. For instance, this holds for the first and third rule of Example 3 and the first, fourth and fifth rule of Example 4.

As heuristics for choosing the interpretations we mention:

- Choose $c_A = 2$ for constants c.
- Concentrate on rules for which the right hand side looks more complicated than the left hand side.
- If the operations have some arithmetical meaning, try to choose an interpretation that slightly exceeds the arithmetical meaning.
- If the operations have a natural precedence, try to choose polynomials with a low degree for the lowest symbols and polynomials with a higher degree for higher symbols.
- If a left hand side of a rule $l \to r$ is of the shape $f(x_1, \ldots, x_n)$ then try to choose $f_A(a_1, \ldots, a_n) = [\![r]\!]_\alpha + 1$ for α defined by $\alpha(x_i) = a_i$ for $i = 1, \ldots, n$.
- For an associative rule $x+(y+z) \to (x+y)+z$ try to choose $+_A$ to be one of the polynomials $X+2Y, XY^2, X+Y^2, XY+Y$; for the reversed associative rule interchange X and Y.
- For a distributive rule $x*(y+z) \to (x*y)+(x*z)$ or an endomorphism rule $f(x+y) \to f(x)+f(y)$ try to choose either $+_A = aX+bY$ for some $a, b \geq 1$, $*_A = XY^2, f_A = X^2$, or $+_A = aX+bY+c$ for some $a, b, c \geq 1$, $*_A = XY, f_A = 2X$.

Exercise 3 Prove termination of the following TRSs by means of polynomials.

- $f(g(x)) \to h(x), \quad h(x) \to g(f(x))$;
-

$$\begin{array}{lcl} 0+x & \to & x \\ x+0 & \to & x \\ x*0 & \to & 0 \\ x*(y+z) & \to & (x*z)+(y*x). \end{array}$$

2.3 Polynomial interpretations modulo AC

Instead of proving termination by applying Theorem 5 to polynomial interpretations we can also prove termination modulo equations by applying Theorem 8. The only extra condition is that the polynomials have to satisfy the equations. Often the equations consist of associativity and commutativity (AC) of one or more binary operators.

Example 5

$$\begin{array}{rcl} 0+x & \to & x \\ s(x)+y & \to & s(x+y). \end{array}$$

This system describes addition on natural numbers as in Example 3. There the interpretation for $+$ was associative and not commutative. However, by choosing $0_A = 2, s_A = X + 1, +_A = XY$ the polynomials $F_{l,r}$ for the rules are $X, Y - 1$, respectively, which are both strictly positive in natural numbers ≥ 2. Since $+_A$ is both associative and commutative now, we have proved termination modulo AC of this system.

As a second example we mention that the rewrite system of Example 4 is terminating modulo AC of the $+$-operator since for $+_A$ an operation was chosen which is both commutative and associative. In fact this was the motivation of this rewrite system, see [8]. There the purpose was to prove completeness of the following equational axiomatization for bisimulation equivalence:

$$\begin{array}{rcl} x+y & = & y+x \\ x+(y+z) & = & (x+y)+z \\ x+x & = & x \\ (x+y)\cdot z & = & (x\cdot z)+(y\cdot z) \\ (x\cdot y)\cdot z & = & x\cdot(y\cdot z) \\ x+\delta & = & x \\ \delta\cdot x & = & \delta. \end{array}$$

If such an equational system can be described by a confluent terminating TRS, then for proving completeness of the axiomatization it suffices to prove that for normal forms the notions of equality and bisimulation equivalence coincide. However, due to commutativity such a TRS does not exist. That's why the system is taken modulo associativity and commutativity of the $+$-operator: then the remaining rules yield the TRS given in example 4 which is terminating modulo AC as we saw, and which is confluent modulo AC since all critical pairs are joinable. Now the proof of completeness of the equational axiomatization can be given by proving that for the normal forms with respect to this TRS the notions of equivalence modulo AC and bisimulation equivalence coincide. A similar approach can be followed for various extensions ([8, 27]); this application of rewrite techniques can be considered as a standard approach for proving completeness of equational axiomatizations.

In these two examples we saw that $X + Y$ and XY are suitable interpretations for AC operators since they obey both commutativity and associativity. On can wonder whether more choices are possible. Indeed there are, like $2XY+3X+3Y+3$. The following characterization is from [11].

Proposition 12 *A polynomial in two variables satisfies commutativity and associativity if and only if it is of the shape* $aXY + b(X + Y) + c$ *with* $ac + b = b^2$.

Proof: If the degree in X is greater than one, then considering the degree in the associativity equality yields a contradiction. Similar for Y, hence the polynomial is of the shape $aXY + bX + dY + c$. Commutativity yields $b = d$. The polynomial $aXY + bX + bY + c$ is associative if and only if

$$(aXY + bX + bY + c)(aZ + b) + bZ + c = (aX + b)(aYZ + bY + bZ + c) + bX + c$$

which is easily checked to be equivalent to $ac + b = b^2$. □

Not all polynomials satisfying the criterion from Proposition 12 are suitable for polynomial interpretations since not all are strictly monotone and well-defined. The next proposition characterizes these extra conditions.

Proposition 13 *A polynomial of the shape* $aXY+bX+bY+c$ *is strictly monotone in both arguments and well-defined in* $A = \{n \in \mathbb{N} \mid n \geq 2\}$ *if and only if* $a \geq 0$, $b \geq 1 - 2a$ *and* $c \geq 2 - 4a - 4b$.

Proof: Let F be such a polynomial. If $a \geq 0$ and $b \geq 1 - 2a$ then $F(x + 1, y) - F(x, y) = ay + b \geq 2a + b \geq 1$, proving strict monotonicity in the first argument, and similar for the second argument. Due to monotonicity for well-definedness it suffices to prove $F(2, 2) \geq 2$, which is equivalent to $c \geq 2 - 4a - 4b$. The converse is similar. □

Exercise 4 Prove that the following single rule TRSs are both terminating modulo AC.

- $s(x + y) \to s(x) + s(y)$;
- $s(s(x)) + x \to s(s(x + s(x)))$.

2.4 Lexicographic combinations

Sometimes, while trying to prove polynomial termination, one achieves interpretations for which the left hand side is not strictly greater than the right hand side, but only greater or equal. This can still be fruitful in proving termination: if the smaller TRS consisting of the rules for which this non-strict inequality holds is terminating, then the whole system can be concluded to be terminating as well. In the general framework of monotone algebras this is stated in the next proposition; for a well-founded monotone algebra $(A, \Sigma_A, <)$ write

$$t \geq_A u \quad \Leftrightarrow \quad \forall \alpha : \mathcal{X} \to A \ : \ [\![t]\!]_\alpha \geq [\![u]\!]_\alpha.$$

Note that $\geq_A$ is essentially more than the union of $>_A$ and equality.

Proposition 14 *Let* (Σ, R) *be a TRS for which* $R = R' \cup R''$ *and let* $(A, \Sigma_A, <)$ *be a well-founded monotone* Σ*-algebra for which*

- $l >_A r$ *for every rule* $l \to r$ *in* R'*, and*
- $l \geq_A r$ *for every rule* $l \to r$ *in* R''*, and*
- (Σ, R'') *is terminating.*

Then (Σ, R) *is terminating.*

Proof: Due to Theorem 5 R'' admits a compatible well-founded monotone algebra $(B, \Sigma_B, \prec)$. Now define $C = A \times B$ with the lexicographic order

$$(a, b) \ll (a', b') \quad \Leftrightarrow \quad a < a' \vee (a = a' \wedge b \prec b'),$$

and for every $f \in \Sigma$:

$$f_C((a_1, b_1), \ldots, (a_n, b_n)) = (f_A(a_1, \ldots, a_n), f_B(b_1, \ldots, b_n))$$

for $a_1, \ldots, a_n \in A, b_1, \ldots, b_n \in B$. For $\gamma : \mathcal{X} \to C$ we can write $\gamma(x) = (\alpha(x), \beta(x))$ for every $x \in \mathcal{X}$, for some $\alpha : \mathcal{X} \to A$, $\beta : \mathcal{X} \to B$. One easily checks

$$[\![t]\!]_\gamma = ([\![t]\!]_\alpha, [\![t]\!]_\beta)$$

for every term t, and hence the well-founded monotone algebra $(C, \Sigma_C, \ll)$ is compatible with R. Due to Theorem 5 now (Σ, R) is terminating. □

Example 6 The following TRS is closely related to the one from Example 3:

$$\begin{array}{rcl} 0 + x & \to & x \\ s(x) + y & \to & s(x + y) \\ 0 * x & \to & 0 \\ s(x) * y & \to & (x * y) + y \end{array}$$

However, the choice $0_A = 2, s_A = X + 3, +_A = 2X + Y, *_A = XY$ now fails for the last rule. It can even be shown that this system is not polynomially terminating: no other choice of polynomials is possible[2]. A termination proof can be given by using elementary functions instead of polynomials: choose $0_A = 2, +_A = 2X + Y, s_A = X + 2$ and $x *_A y = y * 2^x$.

However, using a lexicographic combination there is no reason to leave polynomials: choose $0_A = 2, +_A = X + Y, s_A = X + 2, *_A = XY$, then we get equality for the second rule and strict inequality for the other rules. Hence we can apply Proposition 14 by choosing R'' to consist of the second rule and R' to consist of the rest. Termination of R'' follows from the interpretation $+_A = 2X + Y, s_A = X + 1$, hence termination of the full system follows from Proposition 14.

Clearly Proposition 14 easily extends to n-tupels of TRSs instead of only pairs (R', R'').

2.5 Other examples

We conclude this section by three examples of termination proofs by means of well-founded monotone algebras in which the orders are not total. In Example 11,

[2] A rough sketch of the proof is the following. Write $x +_A y = x * f(x, y) + g(y)$. Taking x constant in the last rule yields $p(y) > q(y) * f(q(y), y) + g(y)$ for two polynomials p, q of the same degree. Considering degrees in this inequality yields that f is a constant. Then the second rule yields that $f > 1$ and $s_A = X + c$ for some constant c. Again take x constant in the last rule, then considering the leading coefficient gives rise to a contradiction.

Proposition 29 and Example 12 we shall prove that choosing non-total orders is essential for these examples. For all three cases we define:

$$A = \{0,1\} \times \mathbb{N}^+ \quad \text{and} \quad (a,n) < (b,m) \quad \Leftrightarrow \quad a = b \wedge n < m;$$

note that $<$ is indeed a well-founded partial order on A which is not total. The overloading of the symbol $<$ does not cause confusion since from the shape of the arguments it is clear which of the two orders is intended.

Example 7 Consider the TRS consisting of the rule:

$$f(f(x)) \ \to \ f(g(f(x))).$$

The intuition is that if f is applied to $f(\cdots)$ we want to obtain a higher weight then if f is applied to $g(\cdots)$. To be able to distinguish between these arguments $f(\cdots)$ and $g(\cdots)$ we let $f_A(\cdots)$ always result in $(1,\ldots)$ and we let $g_A(\cdots)$ always result in $(0,\ldots)$. Define

$$f_A(0,n) = (1,n), \quad f_A(1,n) = (1,n+1), \quad g_A(0,n) = (0,n), \quad g_A(1,n) = (0,n)$$

for all $n \in \mathbb{N}$. Both f_A and g_A are strictly monotone, and

$$\begin{array}{lcllcl} f_A(f_A(0,n)) & = & (1,n+1) & > & (1,n) & = & f_A(g_A(f_A(0,n))), \\ f_A(f_A(1,n)) & = & (1,n+2) & > & (1,n+1) & = & f_A(g_A(f_A(1,n))) \end{array}$$

for all $n \in \mathbb{N}$, proving termination.

Example 8 Consider the TRS with the two rules:

$$\begin{array}{lcl} f(g(x)) & \to & f(f(x)), \\ g(f(x)) & \to & g(g(x)). \end{array}$$

Define

$$f_A(0,n) = (1,2n), \quad f_A(1,n) = (1,n+1),$$
$$g_A(0,n) = (0,n+1), \quad g_A(1,n) = (0,2n).$$

Both f_A and g_A are strictly monotone, while

$$\begin{array}{lcllcl} f_A(g_A(0,n)) & = & (1,2n+2) & > & (1,2n+1) & = & f_A(f_A(0,n)), \\ f_A(g_A(1,n)) & = & (1,4n) & > & (1,n+2) & = & f_A(f_A(1,n)), \\ g_A(f_A(0,n)) & = & (0,4n) & > & (0,n+2) & = & g_A(g_A(0,n)), \\ g_A(f_A(1,n)) & = & (0,2n+2) & > & (0,2n+1) & = & g_A(g_A(1,n)). \end{array}$$

for all $n \in \mathbb{N}$, proving termination.

Example 9 Let the TRS consist of the rule:

$$f(0,1,x) \ \to \ f(x,x,x).$$

The origin of this TRS is the example of [62] showing that termination is not modular, after that it has served as a counter-example for many properties. Define

$$0_A = (0,1), \quad 1_A = (1,1),$$

$$f_A((a,n),(b,m),(c,k)) = \begin{cases} (0,n+m+k) & \text{if } a = b \\ (0,n+m+3k) & \text{if } a \neq b \end{cases}$$

The function f_A is strictly monotone in all three arguments. For all $(a,n) \in A$ we have

$$f_A(0_A, 1_A, (a,n)) = (0, 3n+2) > (0, 3n) = f_A((a,n),(a,n),(a,n)),$$

proving termination.

Exercise 5 Prove that the following single rule TRSs are both terminating.

- $f(g(x)) \to f(s(s(g(x))))$;
- $f(0,x) \to f(s(0),x)$.

3 A hierarchy of termination

In this section we introduce a number of properties of TRSs that are slightly stronger than termination. The most important of them are *simple termination* and *total termination*. We introduce them by restrictions on corresponding monotone algebras but we also give more syntactical characterizations. One motivation for considering these notions is in the observation that most syntactical techniques (to be described in Section 4) not only prove termination, but also simple termination and total termination. As a consequence for a TRS for which by results from the present section it can be seen that it is not simply terminating, or not totally terminating, we know in advance that its termination can not be proved directly by means of that kind of syntactical techniques. Readers only interested in positive results may skip most of this section.

3.1 Simple termination

The basic semantical method of proving termination of a TRS consists of choosing a well-founded monotone algebra compatible with the TRS. If the carrier set of the algebra is chosen to be a well-known set equipped with a well-known well-founded order as we did in our examples, then well-foundedness is no problem. However, if some other carrier set is chosen, for instance a set of terms, and for the order some recursive definition is given, then proving well-foundedness can be a non-trivial issue. In fact this is what happens in the syntactical approach to be discussed in Section 4.

In case of finite signature it is possible to replace the well-foundedness condition in the definitions of a reduction order and a well-founded monotone algebra by a simplicity condition. In recursive definitions this simplicity condition is far easier to verify. First we give the definition.

Definition 15 *A* simple monotone Σ-algebra $(A, \Sigma_A, <)$ *is defined to be a* Σ-*algebra* (A, Σ_A) *for which the set* A *is provided with a partial order* $<$ *such that each algebra operation is strictly monotone in all of its arguments, and*

$$f_A(a_1, \ldots, a_n) \geq a_i$$

for each $f \in \Sigma$, $a_1, \ldots, a_n \in A$, $i \in \{1, \ldots, n\}$.

Definition 16 *For a set* Σ *of operation symbols we define* $Emb(\Sigma)$, *or, shortly,* Emb, *to be the TRS consisting of all the rules*

$$f(x_1, \ldots, x_n) \;\to\; x_i$$

with $f \in \Sigma$ *and* $i \in \{1, \ldots, n\}$ *where* n *is the arity of* f. *These rules are called* embedding rules.

The following propositions state that for proving termination by a simple monotone Σ-algebra, well-foundedness is essentially obtained for free if Σ is finite. The key argument is Kruskal's Tree Theorem([44]); the next proposition can be seen as a reformulation of its basic version.

Proposition 17 *Let Σ be a finite signature. Let $\{u_i\}_{i\in\mathbb{N}}$ be any infinite sequence of ground terms over Σ. Then there is some $i < j$ such that*

$$u_j \to^*_{Emb(\Sigma)} u_i.$$

Proof: It is straightforward how terms can be considered as labelled trees in the terminology of Kruskal's Tree Theorem, where the set of labels corresponds to the signature. Since the signature is finite, the equality relation on the signature is a well-quasi-order. It is easy to check that for this choice of the well-quasi-order the embedding relation in Kruskal's Tree Theorem coincides with $\leftarrow^*_{Emb(\Sigma)}$. Now Kruskal's Tree Theorem states that $\leftarrow^*_{Emb(\Sigma)}$ is a well-quasi-order from which the required result follows from the definition of a well-quasi-order.

A straightforward proof of this finite version of Kruskal's Tree Theorem can be found in [16]. □

Proposition 18 *Let Σ be finite and let $(A, \Sigma_A, <)$ be a simple monotone Σ-algebra. Let $A' \subseteq A$ be the homomorphic image of the ground terms. Then $<$ is well-founded on A'.*

Proof: Let h be the homomorphism from ground terms to A. Assume that $<$ is not well-founded on A'. Since every element of A' is in the image of h there is an infinite sequence

$$h(t_0) > h(t_1) > h(t_2) > h(t_3) > \cdots,$$

Applying Proposition 17 yields $t_j \to^*_{Emb(\Sigma)} t_i$ for some $i < j$. Using the simplicity condition of $(A, \Sigma_A, <)$ it easily follows that $h(t) \geq h(u)$ for ground terms t, u satisfying $t \to_{Emb(\Sigma)} u$. Hence $h(t_j) \geq h(t_i)$, contradicting irreflexivity and transitivity of $<$. □

Proposition 19 *Let Σ be finite and let $(A, \Sigma_A, <)$ be a simple monotone Σ-algebra. Let (Σ, R) be a TRS such that $l >_A r$ for all rewrite rules $l \to r$ of R. Then (Σ, R) is terminating.*

Proof: Apply Theorem 5 and Proposition 18: A' is a well-founded monotone algebra compatible with R. In the case that Σ does not contain constants, add one dummy constant symbol. Choose an arbitrary interpretation in A for this dummy constant. This enforces $A' \neq \emptyset$. □

This proposition does not hold without the restriction of Σ being finite as is shown in the following example.

Example 10 Let Σ consist only of the constants a_i, $i \in \mathbb{N}$, and let R consist of the rules $a_i \to a_{i+1}$ for $i \in \mathbb{N}$. Then clearly R is not terminating, but choosing $A = \mathbb{Z}$, $a_{i,A} = -i$ for $i \in \mathbb{N}$, and the usual order on $\mathbb{Z}$ yields a compatible simple monotone Σ-algebra.

Definition 20 *A TRS is called* simply terminating *if it admits a compatible simple well-founded monotone algebra.*

The above propositions state that for finite Σ the well-foundedness condition can be removed without changing this definition. For infinite Σ this is not true as we saw in Example 10. In [52, 53] an alternative more complicated definition of simple termination is given, which coincides with the definition given here for finite signatures, and also implying well-foundedness in case of infinite signatures.

The following proposition gives a characterization of simple termination not referring to monotone algebras.

Proposition 21 *Let (Σ, R) be a TRS. Then the following assertions are equivalent:*

(1) *R is simply terminating;*
(2) *$R \cup Emb(\Sigma)$ is simply terminating;*
(3) *$R \cup Emb(\Sigma)$ is terminating.*

Proof: The implication (2) $\Rightarrow$ (1) is trivial. For proving (1) $\Rightarrow$ (2) let $(A, \Sigma_A, <)$ be a simple well-founded monotone Σ-algebra compatible with R. Since we allow equality in the definition of simplicity but need strict inequality for compatibility, we have to modify the algebra. We do this by adjoining an extra argument keeping track of the size: we choose

$$B = A \times \mathbb{N}$$

having the lexicographic order

$$(a, k) < (a', k') \quad \Leftrightarrow \quad a < a' \vee (a = a' \wedge k < k').$$

From the arguments of $<$ it should be clear which of the three meanings of $<$ is intended. Define

$$f_B((a_1, k_1), \ldots, (a_n, k_n)) = (f_A(a_1, \ldots, a_n), 1 + \sum_{i=1}^{n} k_i).$$

Now $(B, \Sigma_B, <)$ is a simple well-founded monotone algebra compatible with both R and $Emb(\Sigma)$, proving (2).

The implication (2) $\Rightarrow$ (3) is trivial. Finally, assume that (3) holds. Then according to Theorem 5 there is a well-founded monotone Σ-algebra $(A, \Sigma_A, <)$ compatible with $R \cup Emb(\Sigma)$. Since it is compatible with $Emb(\Sigma)$ it is also a simple well-founded monotone Σ-algebra. This implies (2). $\Box$

Example 11 In Example 7 and Example 9 in Subsection 2.5 we proved that the two single rule TRSs

$$f(f(x)) \ \to \ f(g(f(x)))$$

and

$$f(0, 1, x) \ \to \ f(x, x, x)$$

are terminating. From Proposition 21 we easily see that neither is simply terminating, since adding embedding rules yield the infinite reductions

$$f(f(x)) \ \to \ f(g(f(x))) \ \to_{Emb} \ f(f(x)) \ \to \cdots$$

and

$$\begin{array}{rcl} f(0, 1, f(0, 1, 0)) & \to & f(f(0, 1, 0), f(0, 1, 0), f(0, 1, 0)) \\ & \to_{Emb} & f(0, f(0, 1, 0), f(0, 1, 0)) \\ & \to_{Emb} & f(0, 1, f(0, 1, 0)) \\ & \to & \cdots, \end{array}$$

respectively.

Exercise 6 Prove that the TRSs from Exercise 5 are not simply terminating.

Exercise 7 Prove that the TRS $f(f(g(g(x)))) \to g(g(g(f(f(f(x))))))$ is not simply terminating.

(An extensive analysis of termination of this kind of single rule TRSs, including the challenging proof of termination of this particular copy, is given in [68])

For some kinds of TRSs, however, the notions of termination and simple termination coincide.

Definition 22 *The* size $|t|$ *of a term* t *is defined inductively by*

$$\begin{array}{ll} |x| = |c| = 1 & \textit{for constants } c \textit{ and variables } x, \textit{ and} \\ |f(t_1, \ldots, t_n)| = 1 + \sum_{i=1}^{n} |t_i| & \textit{for function symbols } f. \end{array}$$

A TRS R *is called* size-non-increasing *if* $|t| \geq |u|$ *for all rewrite steps* $t \to_R u$.

Size-non-increasingness of a TRS is immediately established from its rules by the following lemma.

Lemma 23 *A TRS* R *is size-non-increasing if and only if it is non-duplicating and* $|l| \geq |r|$ *for all rules* $l \to r$ *in* R.

Proof: If R is size-non-increasing then by definition $|l| \geq |r|$ for all rules $l \to r$ in R. Substituting a sufficiently big term for a duplicating variable always contradicts size-non-increasingness. Hence if R is size-non-increasing then it is non-duplicating.

Conversely one easily checks that

$$|C[t^\sigma]| = |C[x]| - 1 + |t| + \sum_{x \in Var(t)} n(t,x) * (|x^\sigma| - 1)$$

for all terms t, contexts C and substitutions σ, where $Var(t)$ is the set of variables occurring in t and $n(t,x)$ is the number of occurrences of x in t. Let R be non-duplicating, then $n(r,x) \leq n(l,x)$ for all $x \in Var(l)$ for all rules $l \to r$ in R. Further let $|l| \geq |r|$ for all rules $l \to r$ in R, then for every rewrite step $C[l^\sigma] \to_R C[r^\sigma]$ we obtain

$$|C[l^\sigma]| - |C[r^\sigma]| = |l| - |r| + \sum_{x \in Var(l)} (n(l,x) - n(r,x)) * (|x^\sigma| - 1) \geq 0,$$

proving size-non-increasingness of R. □

Proposition 24 *A size-non-increasing TRS is simply terminating if and only if it is terminating.*

Proof: The 'only if'-part is by definition. For the 'if'-part assume (Σ, R) is any terminating size-non-increasing TRS. Let $A = \mathbb{N}$, let $<$ be the usual order on $\mathbb{N}$ and let $c_A = 1$ for constants c, and $f_A(x_1, \ldots, x_n) = 1 + \sum_{i=1}^{n} x_i$ for function symbols f. Choose $R' = Emb(\Sigma)$, then reading R'' for R all requirements of Proposition 14 are fulfilled, concluding termination of $R \cup Emb(\Sigma)$. By Proposition 21 we conclude that R is simply terminating. □

For example, we conclude that Example 8 in Subsection 2.5 is not only terminating but also simply terminating.

Next we characterize simple termination by means of properties of reduction orders.

Definition 25 *A* simplification order *([16]) is an order $<$ on terms, closed under substitutions and contexts, for which $f(x_1,\ldots,x_n) > x_i$ for all operation symbols f and all $i \in \{1,\ldots,n\}$. A TRS R is called* simplifying *([39]) if there exists a simplification order $<$ such that $t' < t$ for all terms t, t' satisfying $t \to_R t'$.*

Proposition 26 *Let $<$ be a simplification order on $Ter(\Sigma)$.*

(1) *If $u \to^+_{Emb(\Sigma)} t$ then $u > t$.*
(2) *If Σ is finite then $<$ is well-founded.*

Proof: Statement (1) is immediate from the definitions. For (2) assume by contradiction there exists an infinite descending sequence $\{t_i\}_{i\in\mathbb{N}}$ of terms over Σ, i.e., $t_i > t_{i+1}$ for all $i \in \mathbb{N}$, for some simplification order $<$. Assume some t_{i+1} contains a variable x not contained in t_i. Then by choosing $x^\sigma = t_i$ and $y^\sigma = y$ for $y \neq x$ we obtain a contradiction $t_i = t_i^\sigma > t_{i+1}^\sigma = C[t_i]$. So all variables in t_{i+1} are also contained in t_i for all $i \in \mathbb{N}$. Hence only finitely many variables occur in the sequence. By considering these variables as constants, the signature remains finite. Applying Proposition 17 yields $t_j \to^*_{Emb} t_i$ for some $i < j$. Since $t_i > t_j$ by transitivity of $<$, this contradicts (1). □

Now we can extend Proposition 21 in the case of finite signature.

Proposition 27 *Let (Σ, R) be a TRS over a finite signature Σ. Then the following assertions are equivalent:*

(1) *R is simply terminating;*
(2) *$R \cup Emb(\Sigma)$ is terminating;*
(3) *R is simplifying;*
(4) *$\to^+_{R\cup Emb(\Sigma)}$ is irreflexive.*

Proof: The equivalence (1) $\Leftrightarrow$ (2) was proved in Proposition 21 and the implication (2) $\Rightarrow$ (4) is trivial. The implication (4) $\Rightarrow$ (3) follows since $\leftarrow^+_{R\cup Emb(\Sigma)}$ satisfies all requirements of being a simplification order.

It remains to show (3) $\Rightarrow$ (2). Assume (3), then R admits a compatible simplification order $<$. By definition any simplification order is compatible with $Emb(\Sigma)$, hence $<$ is compatible with $R \cup Emb(\Sigma)$. Due to Proposition 26 the simplification order $<$ is a reduction order; by Proposition 3 then $R \cup Emb(\Sigma)$ is terminating, concluding the proof. □

As we saw in Example 10 this proposition does not extend directly to infinite signatures. A first investigation of this kind of problems for infinite signatures has been given in [54]. In [52, 53] the notion of simple termination is revisited in such a way that all these notions coincide, and Proposition 19 can be generalized to infinite signatures. For finite signatures modularity of simple termination has been proved in [45].

3.2 Total termination

If $<_1 \subset <_2$ for two well-founded orders $<_1$ and $<_2$, and $<_1$ succeeds in proving termination of a TRS, then the same holds for $<_2$. Hence the bigger the order is, the more powerful for proving termination. This observation holds for both the semantical view (monotone algebras) and the syntactical view (reduction orders). From both viewpoints it is natural to consider the maximal kind of orders: total orders. It turns out that both the semantical and syntactical notion of 'termination provable by a total order' are equivalent: total termination.

Definition 28 *A TRS* (Σ, R) *is called* totally terminating *if it admits a compatible well-founded monotone* Σ*-algebra* $(A, \Sigma_A, <)$ *for which the order* $<$ *is total on* A*.*

Proposition 29 *Total termination implies simple termination.*

Proof: Let (Σ, R) be a totally terminating TRS and let $(A, \Sigma_A, <)$ be a compatible well-founded monotone Σ-algebra for which the order $<$ is total on A. Assume it is not a simple monotone Σ-algebra. Then there exist $f \in \Sigma$, $a_1, \ldots, a_n \in A$ and $i \in \{1, \ldots, n\}$ such that $f_A(a_1, \ldots, a_n) \geq a_i$ does not hold. From totality we conclude:

$$a_i > f_A(a_1, \ldots, a_n).$$

Define $g : A \to A$ by $g(x) = f_A(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)$, then g is strictly monotone. We obtain an infinite sequence

$$a_i > g(a_i) > g(g(a_i)) > g(g(g(a_i))) > \cdots,$$

contradicting the well-foundedness of $(A, <)$. Hence $(A, \Sigma_A, <)$ is a compatible simple monotone Σ-algebra, proving simple termination. □

If the alternative definition of simple termination is used as it is given in [52, 53], then Proposition 29 does not hold any more for infinite signatures.

For the syntactical counterpart of the definition of total termination it is generally not useful to consider open terms:[3] two variables x, y are never comparable by a reduction order since $x < y$ would imply $x^\sigma < y^\sigma$ for all σ which does not hold if x^σ and y^σ are chosen to be equal. Instead we consider ground terms. The algebra of ground terms over Σ is denoted by $Ter_0(\Sigma)$; for this algebra to be non-empty, we require the existence of at least one constant. This is not a strong restriction: if the original signature Σ does not contain a constant we simply add one, without affecting the definitions we already gave. Remember that a reduction order is defined to be a well-founded order that is closed under subsititutions and contexts; for ground terms there is nothing to substitute by which the requirement of closedness under subsititutions is empty and may be omitted.

To speak about compatibility with a reduction order on $Ter_0(\Sigma)$ we cannot compare l and r for rewrite rules $l \to r$ directly any more since usually these terms are no ground terms. Instead we give the following definition.

Definition 30 *A reduction order* $<$ *on* $Ter_0(\Sigma)$ *is called* compatible *with a TRS* (Σ, R) *if* $l^\sigma > r^\sigma$ *for every ground substitution* σ *and every rewrite rule* $l \to r$ *in* R*.*

Proposition 31 *Let* Σ *be a signature containing at least one constant. Then a TRS* (Σ, R) *is totally terminating if and only if it admits a compatible total reduction order on* $Ter_0(\Sigma)$*.*

Proof: If $<$ is a compatible total reduction order on $Ter_0(\Sigma)$, then $(Ter_0(\Sigma), \Sigma, <)$ is easily checked to be a compatible well-founded monotone algebra. Since $<$ is total, this proves total termination.

Conversely, let $(A, \Sigma_A, <)$ be a compatible well-founded monotone Σ-algebra for which the order $<$ is total on A. For $t \in Ter_0(\Sigma)$ the interpretation $[\![t]\!]_\alpha$ in A does not depend on $\alpha : \mathcal{X} \to A$ and we may omit the subscript α. Choose any total well-founded order $\prec$ on Σ; Zermelo's Theorem (equivalent to the Axiom of Choice) states that such $\prec$ exists for every set Σ. In Proposition 43 we shall see that such

[3] For rewriting systems involving only one variable the existence of a compatible total reduction order on open terms makes sense, but is not equivalent to our notion of total termination. For instance, the TRS consisting of the two rules $f(f(x)) \to g(f(x))$ and $g(g(x)) \to f(g(x))$ is totally terminating but does not admit a compatible total reduction order on open terms.

a total well-founded order $\prec$ on Σ gives rise to a total reduction order $\prec_{rpo}$ on $Ter_0(\Sigma)$. Now one checks that $\ll$ defined by

$$t \ll u \quad \Leftrightarrow \quad [\![t]\!] < [\![u]\!] \vee ([\![t]\!] = [\![u]\!] \wedge t \prec_{rpo} u)$$

is a total reduction order on $Ter_0(\Sigma)$ compatible with the TRS (Σ, R). $\Box$

In order to prove that a particular TRS is not totally terminating we can apply Propositions 21 and 29: if $R \cup Emb$ admits an infinite reduction then R is not totally terminating. This only works if the TRS is not simply terminating. The following theorem can be used to prove that a particular simply terminating TRS is not totally terminating, concluding that standard techniques for proving termination are not applicable.

Proposition 32 *Let (Σ, R) be any TRS for which $C[t] \to^+_{R \cup Emb(\Sigma)} C[u]$ for some context C and terms t, u. Then (Σ, R) is totally terminating if and only if $(\Sigma, R \cup \{t \to u\})$ is a totally terminating TRS.*

Proof: The 'if'-part is by definition. For the 'only if'-part assume that (Σ, R) is totally terminating. Add a constant to Σ if it does not contain one. By Proposition 31 the TRS (Σ, R) admits a compatible total reduction order $<$ on $Ter_0(\Sigma)$. If $x_i^\sigma > (f(x_1, \ldots, x_n))^\sigma$ for any ground substitution σ and any rule $f(x_1, \ldots, x_n) \to x_i$ in the TRS $(\Sigma, Emb(\Sigma))$, then this would give rise to an infinite descending sequence, similar as in the proof of Proposition 29, contradicting well-foundedness. By totality we conclude $(f(x_1, \ldots, x_n))^\sigma > x_i^\sigma$ for any ground substitution σ and any rule $f(x_1, \ldots, x_n) \to x_i$ in $(\Sigma, Emb(\Sigma))$, hence $<$ is compatible with the TRS $(\Sigma, R \cup Emb(\Sigma))$. Let σ be an arbitrary ground substitution. Applying this compatibility to $(C[t])^\sigma \to^+_{R \cup Emb(\Sigma)} (C[u])^\sigma$ yields $C^\sigma[t^\sigma]) = (C[t])^\sigma > (C[u])^\sigma = C^\sigma[u^\sigma])$. Suppose $u^\sigma \geq t^\sigma$, then this contradicts the property that $<$ is closed under contexts. By totality we conclude $t^\sigma > u^\sigma$. This holds for arbitrary σ, hence $<$ is compatible with $(\Sigma, R \cup \{t \to u\})$. From Proposition 31 we conclude that $(\Sigma, R \cup \{t \to u\})$ is totally terminating. $\Box$

Example 12 In Example 8 we saw that the TRS R consisting of the two rules

$$\begin{array}{lcl} f(g(x)) & \to & f(f(x)) \\ g(f(x)) & \to & g(g(x)) \end{array}$$

is terminating. By Proposition 24 it is even simply terminating. Choosing $t = g(x), u = f(x), C = f(\Box)$ the requirements of Proposition 32 are fulfilled. Since $R \cup \{t \to u\})$ admits an infinite reduction $g(f(x)) \to g(g(x)) \to g(f(x)) \to \cdots$, we conclude from Proposition 32 that R is not totally terminating.

Exercise 8 Prove that the TRS consisting of the three rules $g(h(x)) \to f(f(x))$, $f(f(x)) \to g(g(x))$ and $f(g(x)) \to g(h(x))$ is not totally terminating.

A more detailed study of total termination is presented in [22, 26]; generalizations of Proposition 32 are given in [24].

3.3 The hierarchy

Let $(A, \Sigma_A, <)$ be a well-founded monotone algebra. Depending on properties of the corresponding well-founded monotone algebras we propose a hierarchy of types of termination.

Definition 33 *A TRS* (Σ, R) *is called* polynomially terminating *if it is compatible with a well-founded monotone algebra* $(A, \Sigma_A, <)$ *in which* $A = \mathbb{N}$, $<$ *is the usual order on* $\mathbb{N}$ *and* f_A *is a polynomial for all* $f \in \Sigma$.

A TRS (Σ, R) *is called* ω-terminating *if it is compatible with a well-founded monotone algebra* $(A, \Sigma_A, <)$ *in which* $A = \mathbb{N}$ *and* $<$ *is the usual order on* $\mathbb{N}$.

Proving polynomial termination was discussed extensively in Subsection 2.2. Both for polynomial termination and ω-termination for any fixed N we may replace $\mathbb{N}$ by $\{n \in \mathbb{N} \mid n > N\}$, yielding equivalent definitions due to linear transformation.

Proposition 34 *The following implications hold:*

polynomial termination
$\Longrightarrow$ *ω-termination*
$\Longrightarrow$ *total termination*
$\Longrightarrow$ *simple termination*
$\Longrightarrow$ *termination.*

Proof: The implication from total termination to simple termination was given by Proposition 29; the other implications are trivial. □

None of the implications in the hierarchy holds in the reverse direction, not even for single rule string rewriting, as we see in the next proposition.

Proposition 35

(1) *The TRS* $f(g(h(x))) \to g(f(h(g(x))))$
is ω-terminating but not polynomially terminating;
(2) *the TRS* $f(g(x)) \to g(f(f(x)))$
is totally terminating but not ω-terminating;
(3) *the TRS* $f(g(f(x))) \to f(f(f(g(g(x)))))$
is simply terminating but not totally terminating;
(4) *the TRS* $f(f(x)) \to f(g(f(x)))$
is terminating but not simply terminating.

Proof: For the proofs of (1) and (2) we refer to [64]. In Examples 7 and 11 we proved (4). Non-total termination in (3) follows from Proposition 32 and the infinite reduction

$$g(g(f(x))) \to g(f(f(g(g(x))))) \to f(f(\underbrace{g(g(f}(g(g(x)))))))) \to \cdots$$

of the rewrite rule $g(f(x)) \to f(f(g(g(x))))$. Simple termination in (3) will be proved in Example 32. □

The hierarchy of Proposition 35 allows lots of extensions and refinements. For instance, within polynomial termination one can distinguish according to the degrees of the polynomials. Between polynomial and ω-termination one can distinguish according to classes of functions allowed, like the elementary functions from [47], or primitive-recursive functions.

Between ω-termination and total termination one can distinguish according to the order types of the total orders in the monotone algebras. For instance, in [26] it has been shown that essentially the only relevant order types are ordinals of the type ω^α, while for all ordinals $\alpha < \omega + 1$ a TRS has been given that is compatible with a total well-founded monotone algebra of order type ω^α, but not with one having a smaller order type. Note that this way of connecting an ordinal to a

TRS has nothing to do with the order type of orders on terms, or with the kind of functions on natural numbers that are allowed. For instance, the system in Example 16 describing Ackermann's function which is known not to be primitive-recursive, can be proved to be ω-terminating in our terminology.

Between total termination and simple termination one has the notion of *non-self-embeddingness* from [55]: a TRS (Σ, R) is called self-embedding if it admits a reduction of the shape

$$t \to_R^+ u \to_{Emb(\Sigma)}^* t.$$

Clearly such a reduction (called a self-embedding reduction) gives rise to an infinite reduction of $R \cup Emb(\Sigma)$. From Proposition 21 we conclude that simple termination implies non-self-embeddingness. Using Proposition 17 one easily proves that for finite signatures non-self-embeddingness implies termination. Example 10 shows that for this implication finiteness of the signature is essential. For both implications the converse does not hold, even not for single rule string rewriting: the rule $f(f(x)) \to f(g(f(x)))$ is self-embedding and terminating; the rule $f(g(x)) \to h(g(g(f(f(h(x))))))$ is non-self-embedding and not simply terminating.

The notion of termination can be weakened in various ways, including innermost normalization, outermost normalization and weak normalization. It was shown in 1996 by Geser that the single string rewriting rule

$$f(g(f(g(x)))) \to g(f(g(f(f(g(x))))))$$

is weakly normalizing, even innermost normalizing, but not outermost normalizing and hence not terminating.

Another way of weakening termination is *non-loopingness* as described in [69]: a TRS (Σ, R) is called looping if it admits a reduction of the shape $t \to_R^+ C[t^\sigma]$ for some term t, some context C and some substitution σ. Clearly termination implies non-loopingness, on the other hand the TRS consisting of the rules

$$f(g(x)) \to f(h(h(x))), \quad h(g(x)) \to g(h(x)), \quad h(c) \to g(c)$$

is non-terminating and non-looping. In [69] also a one rule TRS and a two rule string rewriting system with these properties are given; for one rule string rewriting it is still open whether the notions of termination and non-loopingness coincide. A further weakening of non-loopingness is *acyclicity*: a TRS (Σ, R) is called cyclic if it admits a reduction of the shape $t \to_R^+ t$ for some term t.

Except for polynomial termination for all properties in the hierarchy undecidability of that property for arbitrary finite TRSs has been proved ([35, 55, 13, 66, 30]), sometimes even for single rewrite rules ([15, 50, 48, 31]). In [30] the stronger result of *relative undecidability* has been proved: for all implications $X \Rightarrow Y$ in the hierarchy (except for X being polynomial termination) the property X is undecidable for finite TRSs satisfying Y. For polynomial termination undecidability is conjectured; it does not follow from Proposition 11 since there the polynomial interpretation is already fixed. In [31] most of these results are extended for single rules.

4 Syntactical methods

From Proposition 3 we recall that any TRS is terminating if and only if it admits a compatible reduction order. Here compatibility means that every left hand side is greater than the corresponding right hand side. Hence if there are finitely many rules and the order is effectively computed, then this compatibility is effectively computed too, yielding a mechanizable method for proving termination In this section we discuss the two main classes of reduction orders suitable for mechanizing termination proofs: the recursive path order and the Knuth-Bendix order. Several

extensions of these orders, in particular of the recursive path order, have been proposed; see [61] for an extensive overview.

For the recursive path order a decision procedure for compatibility with a given finite TRS is straightforward from the definition, for the basic version of the Knuth-Bendix order such a procedure is described in [20]. We concentrate on the general behaviour of these two typical orders. In Section 5 we shall see in various examples how a termination proof of a finite TRS over a finite signature can be given by proving termination of an infinite TRS over an infinite signature by means of the recursive path order. Therefore we do not restrict to finite signatures here. Before giving the definitions we introduce some auxiliary notions.

Definition 36 *Let Σ be any signature. A* precedence *is a (strict) partial order on Σ.*

We restrict to partial orders here; it is possible to generalize to quasi-orders, but most of the power of this generalization is also covered by doing some basic preprocessing on the TRS as we shall see in Subsection 5.1.

In order to be able to compare $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$ by comparing the arguments we sometimes need an order on sequences of terms, like the multiset order or a lexicographic order. This way of lifting orders is called the *status*. We shall need preservation of well-foundedness and some monotonicity requirements for this status; we give the following definition.

Definition 37 *A* status *function τ maps every $f \in \Sigma$ to either* mul *or* lex_π *for some permutation π on $\{1, \ldots, n\}$, where n is the arity of f.*

For a partial order $<$ on terms the partial order $<^{\tau(f)}$ is defined on sequences of length n. If $\tau(f) =$ mul then it describes the multiset extension. More precisely,

$$\langle s_1, \ldots, s_n \rangle <^{\mathrm{mul}} \langle t_1, \ldots, t_n \rangle \quad \Leftrightarrow \quad [s_1, \ldots, s_n] <_{\#} [t_1, \ldots, t_n]$$

where $<_{\#}$ is the usual multiset order. If $\tau(f) = \mathrm{lex}_\pi$ then $<^{\tau(f)}$ describes lexicographic comparison in which argument $\pi(1)$ has the highest priority, argument $\pi(2)$ has the next priority, and so on.

For constants and unary symbols the choice of the status has no effect: for constants there are no arguments at all to compare and for unary symbols f all possible choices of $\tau(f)$ satisfy

$$\langle s \rangle <^{\tau(f)} \langle t \rangle \quad \Leftrightarrow \quad s < t.$$

There is no fundamental reason to restrict to multiset order and lexicographic order, but in practice these two kinds of liftings suffice, while giving a fully general definition of status will be rather complicated.

4.1 Recursive path order

The recursive path order with only multiset status goes back to [17]; a version with lexicographic status was first described in [38].

Theorem 38 *Let Σ be any signature. Let $\prec$ be a well-founded precedence on Σ and let τ be a status on Σ. Then there exists exactly one reduction order $\prec_{rpo}$ on $Ter(\Sigma)$ satisfying*

$$\begin{aligned} s \succ_{rpo} t \quad \Leftrightarrow \quad & s = f(s_1, \ldots, s_n) \text{ and} \\ & \text{(1) } s_i = t \text{ or } s_i \succ_{rpo} t \text{ for some } 1 \leq i \leq n, \text{ or} \\ & \text{(2) } t = g(t_1, \ldots, t_m),\ s \succ_{rpo} t_i \text{ for all } 1 \leq i \leq m, \text{ and either} \\ & \qquad \text{(a) } f \succ g, \text{ or} \\ & \qquad \text{(b) } f = g \text{ and } \langle s_1, \ldots, s_n \rangle \succ_{rpo}^{\tau(f)} \langle t_1, \ldots, t_m \rangle. \end{aligned}$$

The proof of Theorem 38 will be given in Subsection 4.2. Here we assume it holds and discuss its power for giving termination proofs and some other consequences.

Definition 39 *The reduction order $\prec_{rpo}$ implied by Theorem 38 is called the* recursive path order. *In case all symbols have a lexicographic status, it is also called the* lexicographic path order.

Definition 40 *A TRS is called* rpo-terminating *if it is compatible with $\prec_{rpo}$ for some well-founded precedence $\prec$ and a status function τ.*

Since $\prec_{rpo}$ is a reduction order by Theorem 38, we conclude from Proposition 3 that every rpo-terminating TRS is indeed terminating.

Note that well-foundedness of $\prec$ is obtained for free in case the signature is finite. For infinite signatures well-foundedness of the precedence $\prec$ is essential for well-foundedness of $\prec_{rpo}$: if $c_1 \succ c_2 \succ c_3 \succ \cdots$ for constants $c_1, c_2, c_3, \ldots$, then also $c_1 \succ_{rpo} c_2 \succ_{rpo} c_3 \succ_{rpo} \cdots$ by clause (2a). Further note that always $m = n$ in clause (2b).

Given a precedence and a status, Theorem 38 yields a straightforward procedure to check whether $t \succ_{rpo} u$ holds for two given terms t, u or not, since for all recursive calls the sums of the sizes of the arguments is strictly smaller. More precisely, if we want to conclude that $t \succ_{rpo} u$ holds then this is done by the characterization of Theorem 38 by proving $s_i \succ_{rpo} t_i$ for $i = 1, \ldots, k$, for some finite k and terms $s_1, \ldots, s_k, t_1, \ldots, t_k$ satisfying $|s_i|+|t_i| < |t|+|u|$. All clauses of the characterization are of this shape, including the requirements on multiset lifting and lexicographic lifting in clause (2b). Considered as a recursive procedure this will always terminate since in every recursive call the sum of the sizes of the arguments decreases. In a high level functional language implementation of this procedure is hardly more than copying the clauses of the characterization in the right format.

The next observation is that, given a finite TRS, it is decidable whether it is compatible with the reduction order $\prec_{rpo}$ for some precedence and some status. This can be done by checking for $l \succ_{rpo} r$ for all (finitely many) rules $l \to r$ and all (finitely many) precedences and status functions. Much more efficient is starting with the empty precedence, and only extend it by choosing $f \succ g$ when it is forced by clause (2a) in trying to prove $l \succ_{rpo} r$ for a rule $l \to r$. In a similar way the choices for τ are not made before they are forced by the proof obligation. In this way checking for compatibility with some $\prec_{rpo}$ and hence checking for rpo-termination can be done purely automatically and very fast; it is one of the first things to try when for a given TRS a termination proof has to be found.

Before giving some examples we give a useful proposition.

Proposition 41 *Let $t \to^+_{Emb} u$. Then $t \succ_{rpo} u$ for every precedence and every status.*

Proof: By clause (1) we conclude $f(s_1, \ldots, s_n) \succ_{rpo} s_i$ for all $i = 1, \ldots, n$, for all f. Since $\prec_{rpo}$ is closed under contexts we conclude $t \succ_{rpo} u$ for $t \to_{Emb} u$. Now the proposition follows from transitivity of $\prec_{rpo}$. □

This means that for trying to prove rpo-termination, all rules $l \to r$ for which $l \to^+_{Emb} r$ may be ignored.

Example 13 We recall the TRS from Example 3 describing integer arithmetic:

$$\begin{array}{rcl} 0 + x & \to & x \\ s(x) + y & \to & s(x + y) \\ 0 * x & \to & 0 \\ s(x) * y & \to & y + (x * y). \end{array}$$

In order to prove termination we will prove $l \succ_{rpo} r$ for all rules $l \to r$ for a suitable precedence $\prec$.

For the first and the third rule $l \to r$ we conclude $l \succ_{rpo} r$ by Proposition 41, or directly by clause (1).

For the second rule we have to choose $+ \succ s$. Then in order to conclude $s(x) + y \succ_{rpo} s(x + y)$ by clause (2a) it suffices to prove $s(x) + y \succ_{rpo} x + y$. This follows from Proposition 41.

For the fourth rule we have to choose $* \succ +$. Then in order to conclude $s(x) * y \succ_{rpo} y + (x * y)$ by clause (2a) it suffices to prove $s(x) * y \succ_{rpo} y$ and $s(x) * y \succ_{rpo} x * y$, both following from Proposition 41.

Hence by choosing any precedence $\prec$ satisfying $s \prec + \prec *$ we have $l \succ_{rpo} r$ for all rules $l \to r$, proving termination.

Note that clause (2b) and the status function do not play a role in this example. Further note that the same proof also holds if the fourth rule is replaced by $s(x)*y \to (x * y) + y$ as we did in Example 6, where we saw that this causes complications in the proof by means of polynomials.

The next proposition states that the requirement of $s \succ_{rpo} t_i$ in clause (2) may be omitted in some cases of clause (2b).

Proposition 42 *Let Σ be any signature, let $\prec$ be a well-founded precedence on Σ and let τ be a status on Σ. Let s, t be arbitrary terms, then*

$$
\begin{array}{lll}
s \succ_{rpo} t & \Leftrightarrow & s = f(s_1, \ldots, s_n) \text{ and} \\
& & \quad (1)\ s_i = t \text{ or } s_i \succ_{rpo} t \text{ for some } 1 \le i \le n, \text{ or} \\
& & \quad (2)\ t = g(t_1, \ldots, t_m), \text{ and either} \\
& & \qquad (a)\ f \succ g \text{ and } s \succ_{rpo} t_i \text{ for all } 1 \le i \le m, \text{ or} \\
& & \qquad (bmul)\ f = g \text{ and } \tau(f) = \mathrm{mul} \text{ and } \langle s_1, \ldots, s_n \rangle \succ_{rpo}^{\mathrm{mul}} \langle t_1, \ldots, t_m \rangle, \text{ or} \\
& & \qquad (blex)\ f = g \text{ and } \tau(f) = \mathrm{lex}_\pi \text{ and there exists } 1 \le i \le n \text{ satisfying} \\
& & \qquad\qquad s_{\pi(j)} = t_{\pi(j)} \text{ for } 1 \le j < i, \text{ and} \\
& & \qquad\qquad s_{\pi(i)} \succ_{rpo} t_{\pi(i)}, \text{ and} \\
& & \qquad\qquad s \succ_{rpo} t_{\pi(j)} \text{ for } i < j \le n.
\end{array}
$$

Proof: Note that clauses (1) and (2a) are exactly the same as the corresponding clauses in Theorem 38. We will prove now that clause (2b) in Theorem 38 holds if and only if either clause (2bmul) or clause (2blex) holds. The 'only if'-part is immediate. For the 'if'-part first assume that clause (2bmul) holds. In order to pove clause (2b) in Theorem 38 we have to show that $s \succ_{rpo} t_i$ holds for all $i = 1, \ldots, n$. From clause (1) in Theorem 38 we know that $s \succ_{rpo} s_i$ holds for all $i = 1, \ldots, n$, hence

$$\langle s \rangle \succ_{rpo}^{\mathrm{mul}} \langle s_1, \ldots, s_n \rangle \succ_{rpo}^{\mathrm{mul}} \langle t_1, \ldots, t_m \rangle$$

from which we conclude that $s \succ_{rpo} t_i$ holds for all $i = 1, \ldots, n$.

It remains to show that clause (2b) in Theorem 38 in case that clause (2blex) holds. Again it suffices to show that $s \succ_{rpo} t_k$ holds for all $k = 1, \ldots, n$. If $k = \pi(j)$ for $i < j \le n$ this is part of clause (2blex). If $k = \pi(i)$ then $s \succ_{rpo} s_k \succ_{rpo} t_k$ and if $k = \pi(j)$ for $1 \le j < i$ then $s \succ_{rpo} s_k = t_k$. Hence $s \succ_{rpo} t_k$ holds for all $k = 1, \ldots, n$. $\Box$

Example 14 As in Example 2 we consider associativity:

$$f(f(x, y), z) \ \to \ f(x, f(y, z)).$$

Since only one operation symbol is involved, the precedence does not play a role here. Choose $\tau(f) = \mathrm{lex}_\pi$ for π being the identity on $\{1, 2\}$. For proving $f(f(x, y), z) \succ_{rpo}$

$f(x, f(y, z))$ it suffices by choosing $i = 1$ in clause (2blex) to prove $f(x, y) \succ_{rpo} x$ and $f(f(x, y), z) \succ_{rpo} f(y, z)$. Both follow immediately from Proposition 41. So the TRS is compatible with $\prec_{rpo}$, and hence terminating.

Example 15 Consider another variant of a TRS describing integer arithmetic:

$$\begin{array}{rcl} 0 + x & \to & x \\ s(x) + y & \to & s(x + y) \\ 0 * x & \to & 0 \\ s(x) * y & \to & y + (y * x). \end{array}$$

For the first three rules $l \to r$ we proved $l \succ_{rpo} r$ for any precedence satisfying $+ \succ s$ in Example 13.

For the fourth rule again choose $* \succ +$. In order to conclude $s(x) * y \succ_{rpo} y + (y * x)$ by clause (2a) we have to prove that $s(x) * y \succ_{rpo} y$ and $s(x) * y \succ_{rpo} y * x$. The first follows from Proposition 41, for the second we choose $\tau(*) = \mathrm{mul}$ and by clause (2bmul) it remains to prove that $\langle s(x), y\rangle \succ_{rpo}^{\mathrm{mul}} \langle y, x\rangle$. This follows from $y = y$ and $s(x) \succ_{rpo} x$.

Hence by choosing any precedence $\prec$ satisfying $s \prec + \prec *$ and $\tau(*) = \mathrm{mul}$ we have $l \succ_{rpo} r$ for all rules $l \to r$, proving termination.

Example 16 The following TRS describes computation of Ackermann's function, which is known not to be primitive-recursive.

$$\begin{array}{rcl} a(0, x) & \to & s(x) \\ a(s(x), 0) & \to & a(x, s(0)) \\ a(s(x), s(y)) & \to & a(x, a(s(x), y)). \end{array}$$

Termination of this system follows from compatibility with $\prec_{rpo}$, by choosing $a \succ s$ and $\tau(a) = \mathrm{lex}_\pi$ for π being the identity on $\{1, 2\}$. Although the TRS is terminating, small terms can have extremely long reductions. For instance, $a(s(s(s(s(s(0))))), s(0))$ reduces to its normal form of the shape $s^n(0)$ where both n and the length of the reduction are numbers of far more than 10^{1000} digits.

Exercise 9 Prove termination of the following three TRSs by means of recursive path order.

$$f(g(x)) \to a(g(f(x)), x);$$

$$\begin{cases} a(f(x), y) & \to & f(a(x, f(y))) \\ a(c, x) & \to & f(x) \\ b(x, f(y)) & \to & a(x, b(y, x)) \\ b(x, c) & \to & x; \end{cases}$$

$$\begin{cases} f(0) & \to & 0 \\ f(s(x)) & \to & s(s(f(x))) \\ g(0) & \to & f(s(0)) \\ g(s(x)) & \to & f(g(x)). \end{cases}$$

From Propositions 21 and 41 we conclude that rpo-termination implies simple termination. It was shown in [24, 22] that rpo-termination implies total termination. If the TRS is finite then rpo-termination implies ω-termination; if moreover all function symbols have multiset status then for the corresponding interpretations in the naturals even primitive-recursive functions can be chosen ([33]).

The latter result does not extend to infinite TRSs. Consider for example the TRS R consisting of the rules

$$g(f(x)) \to \underbrace{f(\cdots(f}_{n}(g(x)))\cdots)$$

for all $n \in \mathbb{N}$. Now rpo-termination of R follows by choosing the precedence $g \succ f$. Conversely, assume that R is ω-terminating. Then there exist strictly increasing functions $f, g: \mathbb{N} \to \mathbb{N}$ satisfying $g(f(x)) > f^n(g(x))$ for all $n, x \in \mathbb{N}$. From $g(f(x)) > g(x)$ one concludes $f(x) > x$, for all $x \in \mathbb{N}$. Hence $g(f(0)) > f^n(g(0)) > f^{n-1}(g(0)) > \cdots > f(g(0))$ for all $n \in \mathbb{N}$, which is impossible. Hence R is not ω-terminating.

However, rpo-termination does not imply polynomial termination, not even for one-rule string rewriting systems. As an example we mention $f(g(h(x))) \to g(f(h(g(x))))$. From Proposition 35 we recall that this TRS is not polynomially terminating, while rpo-termination can be shown by the precedence $f \succ g \succ h$.

Conversely, neither ω-termination nor polynomial termination imply rpo-termination: the TRS consisting of the rules $f(f(x)) \to g(x), g(x) \to f(x)$ is not rpo-terminating, while it admits the very simple polynomial interpretation $f(x) = x + 2$, $g(x) = x + 3$. A single string rewriting rule having the same property is $f(g(f(x))) \to g(f(f(h(x))))$ with the polynomial interpretation $f(x) = 2x$, $g(x) = x + 1$, $h(x) = x$.

Proposition 43 *Let $\prec$ be a well-order on Σ and let $\tau(f) = \mathrm{lex}_\pi$ for all $f \in \Sigma$ for suitable π, i.e., all symbols have lexicographic status. Then $\prec_{rpo}$ restricted to $Ter_0(\Sigma)$ is a well-order.*

Proof: Since $\prec_{rpo}$ is a reduction order by Theorem 38 it is well-founded, and it remains to show totality. This is done by proving the following assertion by induction on n:

> If $t \neq u$ and $|t| + |u| < n$ for $t, u \in Ter_0(\Sigma)$ then either $t \succ_{rpo} u$ or $u \succ_{rpo} t$.

Let $t = f(t_1, \ldots, t_n)$ and $u = g(u_1, \ldots, u_m)$. If $t_i = u$ or $t_i \succ_{rpo} u$ for some $i = 1, \ldots, n$ then from clause (1) we conclude $t \succ_{rpo} u$. If $u_j = t$ or $u_j \succ_{rpo} t$ for some $j = 1, \ldots, m$ then from clause (1) we conclude $u \succ_{rpo} t$. In both these cases we are done. In the remaining case we conclude from the induction hypothesis that $u \succ_{rpo} t_i$ for all $i = 1, \ldots, n$ and $t \succ_{rpo} u_j$ for all $j = 1, \ldots, m$. If $f \succ g$ then $t \succ_{rpo} u$ by clause (2a); if $g \succ f$ then $u \succ_{rpo} t$ by clause (2a). In both cases we are done; since $\prec$ is assumed to be total in the remaining case we have $f = g$. Since $t \neq u$ there exists i for which $t_i \neq u_i$. Let $\tau(f) = \mathrm{lex}_\pi$ and let k be the smallest value ≥ 1 for which $t_{\pi(k)} \neq u_{\pi(k)}$. By the induction hypothesis we either have $t_{\pi(k)} \succ_{rpo} u_{\pi(k)}$ or $u_{\pi(k)} \succ_{rpo} t_{\pi(k)}$. The first case yields $t \succ_{rpo} u$ by clause (2b), the second case yields $u \succ_{rpo} t$ by clause (2b). In all cases we are done. □

4.2 Justification of recursive path order

In this subsection we prove Theorem 38. As far as we know the first full treatment was given in [22], which is roughly followed here. Surprisingly, in the literature where recursive path order is introduced ([17, 18]) a characterization as in Theorem 38 is posed as being a recursive definition, followed by a case analysis for verifying transitivity and irreflexivity. However, there it is left unclear what is meant by a multiset lifting or lexicographic lifting of a relation which is not yet known to be transitive or irreflexive. Only in the unpublished note [38] some incomplete hints for justification are given.

We will construct $\prec_{rpo}$ as being the least fixed point of a continuous operator Φ. Readers familiar with lattices or complete partial orders will recognize $\vee_{n \geq 0} \Phi^n(\bot)$ as being the standard construction for such a least fixed point. In our case $\vee$ corresponds to ordinary set union and $\bot$ corresponds to $\emptyset$. We will present the

proof independent from the theory of complete partial orders. Also the proof of well-foundedness will be given independent from Kruskal's Tree theorem, although using a similar minimal bad sequence argument.

First we collect some required properties of the status function.

Proposition 44 *Let $n \in \mathbb{N}$. Let $\tau = \mathrm{mul}$ or $\tau = \mathrm{lex}_\pi$ for some permutation π on $\{1,\ldots,n\}$, where n is the arity of f. Write $<, <_i$ for partial orders on a fixed set T, for $i \in \mathbb{N}$. Then*

(1) If $\langle t_1,\ldots,t_n\rangle <_1^\tau \langle s_1,\ldots,s_n\rangle$ and $(t_i <_1 s_j \Rightarrow t_i <_2 s_j)$ for all $i,j = 1,\ldots,n$, then $\langle t_1,\ldots,t_n\rangle <_2^\tau \langle s_1,\ldots,s_n\rangle$.

(2) If $t_i < s_i$ for some $i = 1,\ldots,n$ and $t_j = s_j$ for all $j \neq i$ then $\langle t_1,\ldots,t_n\rangle <^\tau \langle s_1,\ldots,s_n\rangle$.

(3) Let $f : T \to T$ be strictly monotone with respect to $<$, i.e., $f(t) < f(s)$ for all t,s satisfying $t < s$. If $\langle t_1,\ldots,t_n\rangle <^\tau \langle s_1,\ldots,s_n\rangle$ then $\langle f(t_1),\ldots,f(t_n)\rangle <^\tau \langle f(s_1),\ldots,f(s_n)\rangle$.

(4) If $<_1 \subseteq <_2 \subseteq <_3 \subseteq \cdots$ then

$$(\textstyle\bigcup_{i=1}^\infty <_i)^\tau = \bigcup_{i=1}^\infty (<_i^\tau).$$

(5) If $S \subseteq T$ and $<$ restricted to S is well-founded, then $<^\tau$ restricted to S^n is well-founded too.

Proof: All properties are easily checked by using the definitions of lexicographic order and multiset order, and their standard properties. Note that the $\supseteq$-part of the continuity condition (4) follows from the monotonicity condition (1). □

For the rest of this subsection we fix Σ to be any signature, $\prec$ to be any precedence on Σ and τ to be any status on Σ.

Theorem 38 consists of an existence claim and uniqueness claim. Uniqueness is stated in the next proposition.

Proposition 45 *Let $\prec_{rpo}^1$ and $\prec_{rpo}^2$ be two orders on $Ter(\Sigma)$ both satisfying the requirements of Theorem 38. Then $\prec_{rpo}^1 = \prec_{rpo}^2$.*

Proof: By symmetry it suffices to prove that $s \succ_{rpo}^2 t$ if $s \succ_{rpo}^1 t$. This is done by induction on the size: as the induction hypothesis we assume that $s' \succ_{rpo}^1 t' \Rightarrow s' \succ_{rpo}^2 t'$ for all s', t' satisfying $|s'| + |t'| < |s| + |t|$. Write $s = f(s_1,\ldots,s_n)$.

If $s \succ_{rpo}^1 t$ by case (1) then either $s_i = t$ or $s_i \succ_{rpo}^1 t$ for some $i = 1,\ldots,n$. Applying the induction hypothesis yields that either $s_i = t$ or $s_i \succ_{rpo}^2 t$. By case (1) we conclude that $s \succ_{rpo}^2 t$.

If $s \succ_{rpo}^1 t$ by case (2) then we can write $t = g(t_1,\ldots,t_m)$. For all $i = 1,\ldots,m$ we have $s \succ_{rpo}^1 t_i$; by the induction hypothesis we conclude that $s \succ_{rpo}^2 t_i$ for all $i = 1,\ldots,m$. In case (2a) we have $f \succ g$ and we conclude $s \succ_{rpo}^2 t$ by case (2a) and we are done.

In the remaining case (2b) we have $f = g$ and $\langle s_1,\ldots,s_n\rangle \succ_{rpo}^{1\tau(f)} \langle t_1,\ldots,t_m\rangle$. From the induction hypothesis we conclude $(s_i \succ_{rpo}^1 t_j \Rightarrow s_i \succ_{rpo}^2 t_j)$ for all $i,j = 1,\ldots,n$; from Proposition 44, part (1) we then conclude $\langle s_1,\ldots,s_n\rangle \succ_{rpo}^{2\tau(f)} \langle t_1,\ldots,t_m\rangle$. We already saw that $s \succ_{rpo}^2 t_i$ for all $i = 1,\ldots,m$, hence $s \succ_{rpo}^2 t$ by case (2b). □

For proving Theorem 38 it remains to show the existence claim. This will be done by giving a definition for $\prec_{rpo}$ and showing that it satisfies all required properties.

Definition 46 *Let $\sqsubset$ be a partial order on $Ter(\Sigma)$. Then the binary relation $\Phi(\sqsubset)$ on $Ter(\Sigma)$ is recursively defined by:*

$$\begin{aligned} t\ \Phi(\sqsubset)\ s \quad \Leftrightarrow \quad & s = f(s_1,\ldots,s_n) \text{ and} \\ & \text{(1) } t = s_i \text{ or } t\ \Phi(\sqsubset)\ s_i \text{ for some } 1 \le i \le n, \text{ or} \\ & \text{(2) } t = g(t_1,\ldots,t_m),\ t_i\ \Phi(\sqsubset)\ s \text{ for all } 1 \le i \le m, \text{ and either} \\ & \qquad \text{(a) } g \prec f, \text{ or} \\ & \qquad \text{(b) } g = f \text{ and } \langle t_1,\ldots,t_m\rangle \sqsubset^{\tau(f)} \langle s_1,\ldots,s_n\rangle. \end{aligned}$$

Since in this definition of $t\ \Phi(\sqsubset)\ s$ the recursive calls $t'\ \Phi(\sqsubset)\ s'$ all satisfy $|s'|+|t'| < |s|+|t|$, as a binary relation $\Phi(\sqsubset)$ is well-defined. Next we prove that it is a partial order again.

Proposition 47 *Let $\sqsubset$ be any partial order on $Ter(\Sigma)$. Then $\Phi(\sqsubset)$ is a partial order on $Ter(\Sigma)$ too.*

Proof: We have to prove irreflexivity and transitivity. First we prove transitivity by induction on the size of the terms. More precisely, for $t\ \Phi(\sqsubset)\ s$ and $s\ \Phi(\sqsubset)\ u$ we shall prove $t\ \Phi(\sqsubset)\ u$, where as the induction hypothesis it is assumed that $(t'\ \Phi(\sqsubset\ )\ s' \wedge s'\ \Phi(\sqsubset)\ u') \Rightarrow t'\ \Phi(\sqsubset)\ u'$ for all t', s', u' satisfying $|t'|+|s'|+|u'| < |t|+|s|+|u|$. Write $s = f(s_1,\ldots,s_n)$ and $u = h(u_1,\ldots,u_k)$. We do the following case analysis.

- Let $s\ \Phi(\sqsubset)\ u$ be by case (1). If $s = u_i$ for some i we have $t\ \Phi(\sqsubset)\ u_i$; if $s\ \Phi(\sqsubset)\ u_i$ we apply the induction hypothesis to t, s, u_i also yielding $t\ \Phi(\sqsubset)\ u_i$. In both cases we have $t\ \Phi(\sqsubset)\ u_i$ from which $t\ \Phi(\sqsubset)\ u$ is concluded by (1).

- Let $s\ \Phi(\sqsubset)\ u$ by case (2) and $t\ \Phi(\sqsubset)\ s$ be by case (1). From the first assumption we conclude $t = s_i$ or $t\ \Phi(\sqsubset)\ s_i$ for some $i = 1,\ldots,n$, from the second we conclude $s_i\ \Phi(\sqsubset)\ u$. Applying the induction hypothesis on t, s_i, u yields $t\ \Phi(\sqsubset)\ u$.

- Let $s\ \Phi(\sqsubset)\ u$ and $t\ \Phi(\sqsubset)\ s$ be both by case (2), write $t = g(t_1,\ldots,t_m)$. Since $t_i\ \Phi(\sqsubset)\ s$ for all $i = 1,\ldots,m$ we can apply the induction hypothesis on t_i, s, u yielding $t_i\ \Phi(\sqsubset)\ u$ for all $i = 1,\ldots,m$. If $g \prec f$ or $f \prec h$ then $g \prec h$ yielding $t\ \Phi(\sqsubset)\ u$ by case (2a). In the remaining case we have $g = f$ and $\langle t_1,\ldots,t_m\rangle \sqsubset^{\tau(f)} \langle s_1,\ldots,s_n\rangle$ and $f = h$ and $\langle s_1,\ldots,s_n\rangle \sqsubset^{\tau(f)} \langle u_1,\ldots,u_k\rangle$. From transitivity of $\sqsubset^{\tau(f)}$ we conclude $t\ \Phi(\sqsubset)\ u$ by case (2b).

Hence $\Phi(\sqsubset)$ is transitive.

For proving irreflexivity we again apply induction on the size of the terms. As the induction hypothesis we assume $\neg(s'\ \Phi(\sqsubset)\ s')$ for terms s' satisfying $|s'| < |s|$, assume by contradiction $s\ \Phi(\sqsubset)\ s$. Write $s = f(s_1,\ldots,s_n)$. Again we do case analysis.

- Let $s\ \Phi(\sqsubset)\ s$ be by case (1). Then $s = s_i$ or $s\ \Phi(\sqsubset)\ s_i$ for some $i = 1,\ldots,n$. Since $s_i\ \Phi(\sqsubset)\ s$ by case (1), we conclude $s_i\ \Phi(\sqsubset)\ s_i$ by transitivity of $\Phi(\sqsubset)$, contradicting the induction hypothesis.

- Let $s\ \Phi(\sqsubset)\ s$ be by case (2a). This contradicts irreflexivity of $\prec$.

- Let $s\ \Phi(\sqsubset)\ s$ be by case (2b). This contradicts irreflexivity of $\sqsubset^{\tau(f)}$.

Hence $\Phi(\sqsubset)$ is irreflexive. $\Box$

Example 17 Let Σ consist of two constants a, b and a unary symbol f. Let $\sqsubset$ be defined by

$$s \sqsubset t \quad \Leftrightarrow \quad s = f^n(a) \wedge t = f^m(b) \text{ for some } m, n \ge 0.$$

Let $\prec$ be defined by $b \prec a \prec f$. Then $\Phi(\sqsubset)$ restricted to ground terms is the following linear order:

$$b\ \Phi(\sqsubset)\ a\ \Phi(\sqsubset)\ f(a)\ \Phi(\sqsubset)\ f(f(a))\ \Phi(\sqsubset)\ f(f(f(a)))\cdots$$

$$\Phi(\sqsubset)\ f(b)\ \Phi(\sqsubset)\ f(f(b))\ \Phi(\sqsubset)\ f(f(f(b)))\cdots.$$

Note that $\sqsubset$ is closed under contexts, but $\Phi(\sqsubset)$ is not. Note that $\sqsubset$ is not contained in $\Phi(\sqsubset)$.

Proposition 48 *Let $\sqsubset_1 \subseteq \sqsubset_2$. Then $\Phi(\sqsubset_1) \subseteq \Phi(\sqsubset_2)$.*

Proof: Again we do induction on the size of the terms: we assume $t\ \Phi(\sqsubset_1)\ s$ and we have to prove $t\ \Phi(\sqsubset_2)\ s$. As the induction hypothesis we assume $(t'\ \Phi(\sqsubset_1)\ s') \Rightarrow (t'\ \Phi(\sqsubset_2)\ s')$ for all s', t' satisfying $|s'| + |t'| < |s| + |t|$.

If $t\ \Phi(\sqsubset_1)\ s$ is by case (1) or (2a) then $t\ \Phi(\sqsubset_2)\ s$ is immediate from the induction hypothesis. If $t\ \Phi(\sqsubset_1)\ s$ is by case (2b) then we still have to prove

$$\langle t_1, \ldots, t_m\rangle \sqsubset_1^{\tau(f)} \langle s_1, \ldots, s_n\rangle \implies \langle t_1, \ldots, t_m\rangle \sqsubset_2^{\tau(f)} \langle s_1, \ldots, s_n\rangle.$$

This follows from Proposition 44, part (1). □

The smallest partial order on $Ter(\Sigma)$ is $\emptyset$, the empty relation. By applying Φ on $\emptyset$ we obtain a new partial order, on which Φ can be applied again, and so on. We define $\prec_{rpo}$ to be the limit of this process as follows.

Definition 49 *Let $\prec_0 = \emptyset$ and $\prec_{i+1} = \Phi(\prec_i)$ for $i \geq 0$. We define*

$$\prec_{rpo} = \bigcup_{i=0}^{\infty} \prec_i .$$

In other words, $t \prec_{rpo} s$ if and only if $t \prec_i s$ for some positive integer i. In order to prove Theorem 38 we have to verify that $\prec_{rpo}$ defined in this way is indeed a reduction order satisfying the given requirements.

Proposition 50 *For all $0 \leq k \leq n$ we have $\prec_k \subseteq \prec_n$.*

Proof: We prove $\prec_k \subseteq \prec_{k+1}$ by induction on k. For $k = 0$ it is trivial; for $k > 0$ it follows from the induction hypothesis and Proposition 48. □

Proposition 51 *The relation $\prec_{rpo}$ is a partial order on $Ter(\Sigma)$.*

Proof: We have to prove irreflexivity and transitivity. First assume $s \prec_{rpo} s$. Then there is some positive integer i with $s \prec_i s$, contradicting irreflexivity of $\prec_i$.

Next assume $t \prec_{rpo} s$ and $s \prec_{rpo} u$. Then there are positive integers i, j with $t \prec_i s$ and $s \prec_j u$. Let k be the maximum of i and j; from Proposition 50 we conclude $t \prec_k s$ and $s \prec_k u$. From transitivity of $\prec_k$ we conclude $t \prec_k u$, hence $t \prec_{rpo} u$. Hence $\prec_{rpo}$ is transitive. □

Proposition 52 *The partial order $\prec_{rpo}$ on $Ter(\Sigma)$ is closed under contexts.*

Proof: Assume $t \prec_{rpo} s$; we have to prove

$$h(u_1, \ldots, u_{i-1}, t, u_{i+1}, \ldots, u_n) \prec_{rpo} h(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_n).$$

Since $t \prec_{rpo} s$ there exists i with $t \prec_i s$. It suffices to prove that

$$h(u_1, \ldots, u_{i-1}, t, u_{i+1}, \ldots, u_n) \prec_{i+1} h(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_n)$$

according to case (2b) in the definition of Φ. For $j = 1, \ldots, n$, $j \neq i$ we have

$$u_j \prec_{i+1} h(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_n)$$

according to case (1). Since $t \prec_{i+1} s$ according to Proposition 50 we have $t \prec_{i+1} h(u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_n)$ according to case (1) in the definition of Φ. It remains to show

$$\langle u_1, \ldots, u_{i-1}, t, u_{i+1}, \ldots, u_n \rangle \prec_i^{\tau(h)} \langle u_1, \ldots, u_{i-1}, s, u_{i+1}, \ldots, u_n \rangle.$$

This follows from Proposition 44, part (2). □

Proposition 53 *The partial order $\prec_{rpo}$ on $Ter(\Sigma)$ is closed under substitutions.*

Proof: Let σ be any substitution. Assume $t \prec_{rpo} s$; we have to prove $t^\sigma \prec_{rpo} s^\sigma$. We do this by proving $t^\sigma \prec_k s^\sigma$ for t, s satisfying $t \prec_k s$, for every $k > 0$. The proof is given by induction on both the size of the terms and k, more precisely, as the induction hypothesis we assume that $t' \prec_k s' \Rightarrow t'^\sigma \prec_k s'^\sigma$ for all s', t' satisfying $|s'| + |t'| < |s| + |t|$, and $t' \prec_{k-1} s' \Rightarrow t'^\sigma \prec_{k-1} s'^\sigma$ for all s', t'.

Again write $s = f(s_1, \ldots, s_n)$. If $t \prec_k s$ is by case (1) in the definition of Φ then either $t = s_i$ or $t \prec_k s_i$ for some $i = 1, \ldots, n$. From the induction hypothesis we conclude that either $t^\sigma = s_i^\sigma$ or $t^\sigma \prec_k s_i^\sigma$, hence $t^\sigma \prec_k s^\sigma$ by case (1).

If $t \prec_k s$ is by case (2) in the definition of Φ then $t = g(t_1, \ldots, t_m)$ and $t_i \prec_k s$ for all $i = 1, \ldots, m$. From the induction hypothesis we conclude that $t_i^\sigma \prec_k s^\sigma$ for all $i = 1, \ldots, m$. For case (2a) we may conclude $t^\sigma \prec_k s^\sigma$ by case (2a). For case (2b) we have $m = n$ and

$$\langle t_1, \ldots, t_m \rangle \prec_{k-1}^{\tau(f)} \langle s_1, \ldots, s_n \rangle.$$

It remains to prove

$$\langle t_1^\sigma, \ldots, t_m^\sigma \rangle \prec_{k-1}^{\tau(f)} \langle s_1^\sigma, \ldots, s_n^\sigma \rangle.$$

This follows from Proposition 44, part (3), using that strict monotonicity of σ with respect to $\prec_{k-1}$ is part of the induction hypothesis. □

Proposition 54 *The partial order $\prec_{rpo}$ on $Ter(\Sigma)$ is a simplification order.*

Proof: By Propositions 51, 52 and 53, $\prec_{rpo}$ is an order which is closed under contexts and substitutions. By case (1) in the definition of Φ we have $f(x_1, \ldots, x_n) \succ_1 x_i$ for all operation symbols f and all $i \in \{1, \ldots, n\}$. Hence also $f(x_1, \ldots, x_n) \succ_{rpo} x_i$. □

Proposition 55 *The partial order $\prec_{rpo}$ on $Ter(\Sigma)$ is a reduction order.*

Proof: Due to Propositions 52 and 53 the only thing to prove is that $\prec_{rpo}$ is well-founded.

If Σ is finite this is immediate from Propositions 26 and 54. Here we give a proof independent of Kruskal's Tree Theorem (as it was used in Proposition 26) that also holds if Σ is infinite. The key argument is similar to the minimal bad sequence argument in the proof of Kruskal's Tree Theorem.

Assume by contradiction that $\prec_{rpo}$ admits an infinite descending sequence. We call a term *well-founded* if it does not occur in an infinite descending sequence. Hence we assume there is some non-well-founded term. Among all non-well-founded terms we choose a term t_0 of minimal size. Since t_0 is non-well-founded there exists a non-well-founded term t_1 satisfying $t_0 \succ_{rpo} t_1$; among all such terms we choose t_1 of minimal size. This process is extended to the following inductive definition: for every $i > 0$ we choose t_i to be a term of minimal size among all non-well-founded terms satisfying $t_{i-1} \succ_{rpo} t_i$. In this way we obtain a particular infinite descending sequence $t_0 \succ_{rpo} t_1 \succ_{rpo} t_2 \succ_{rpo} \cdots$. Due to the definition of $\prec_{rpo}$ we can write $t_i = f_i(u_{i,1}, \ldots, u_{i,n(i)})$ and $t_{i+1}\ \Phi(\prec_{k(i)})\ t_i$, for all $i \in \mathbb{N}$ and suitable $k(i) \geq 0$. First observe that all $u_{i,j}$ are well-founded: since $t_{i-1} \succ_{rpo} u_{i,j}$ for $i > 0$, non-well-foundedness of $u_{i,j}$ contradicts minimality in the definition of t_i.

If $t_{i+1}\ \Phi(\prec_{k(i)})\ t_i$ is by case (1) in the definition of Φ for some i then there exists j such that $t_{i+1} = u_{i,j}$ or $t_{i+1}\ \Phi(\prec_{k(i)})\ u_{i,j}$, contradicting the fact that t_{i+1} is non-well-founded and $u_{i,j}$ is well-founded. Hence all steps $t_{i+1}\ \Phi(\prec_{k(i)})\ t_i$ are by case (2) in the definition of Φ. Hence $f_i \succ f_{i+1}$ or $f_i = f_{i+1}$ for all $i \geq 0$. Since $\prec$ is well-founded, there exists $N \in \mathbb{N}$ such that $f_i = f_{i+1}$ for all $i \geq N$. Hence for $i \geq N$ all steps $t_{i+1}\ \Phi(\prec_{k(i)})\ t_i$ are by case (2b) in the definition of Φ, i.e., $\langle u_{i+1,1}, \ldots, u_{i+1,n}\rangle \prec_{k(i)}^{\tau(f)} \langle u_{i,1}, \ldots, u_{i,n}\rangle$ for all $i \geq N$, where $f = f_N$ and n is the arity of f. By Proposition 44, part (4), we conclude

$$\langle u_{i+1,1}, \ldots, u_{i+1,n}\rangle \prec_{rpo}^{\tau(f)} \langle u_{i,1}, \ldots, u_{i,n}\rangle$$

for all $i \geq N$, contradicting Proposition 44, part (5), where the set S is defined by

$$S = \bigcup_{i=N}^{\infty}(\bigcup_{j=1}^{n} u_{i,j}).$$

□

Now the proof of Theorem 38 is completed by the following proposition.

Proposition 56 *Two terms s, t satisfy $s \succ_{rpo} t$ if and only if $s = f(s_1, \ldots, s_n)$ and*

(1) $s_i = t$ or $s_i \succ_{rpo} t$ for some $1 \leq i \leq n$, or
(2) $t = g(t_1, \ldots, t_m)$, $s \succ_{rpo} t_i$ for all $1 \leq i \leq m$, and either
(a) $f \succ g$, or
(b) $f = g$ and $\langle s_1, \ldots, s_n\rangle \succ_{rpo}^{\tau(f)} \langle t_1, \ldots, t_m\rangle$.

Proof: First we prove the 'if'-part. If we are in case (1) we have $s_i = t$ or $s_i \succ_k t$ for some $1 \leq i \leq n$ and some $k \in \mathbb{N}$. We conclude $s \succ_{k+1} t$ by case (1) of the definition of Φ, hence $s \succ_{rpo} t$. If we are in case (2) we have $s \succ_{k_i} t_i$ for all $1 \leq i \leq m$, for some $k_1, \ldots, k_m$. Let k be the maximum of all k_i. In case (2a) we conclude $s \succ_{k+1} t$ by case (2a) of the definition of Φ, hence $s \succ_{rpo} t$. In case (2b) we apply Proposition 44, part (4), yielding some p such that

$$\langle s_1, \ldots, s_n\rangle \succ_{p}^{\tau(f)} \langle t_1, \ldots, t_m\rangle.$$

Taking q to be the maximum of k and p we obtain $s \succ_{q+1} t$ by case (2b) of the definition of Φ, hence again $s \succ_{rpo} t$.

Next we prove the 'only if'-part; we assume $s \succ_{rpo} t$. Hence for some k we have $t \prec_k s$. Since $\prec_0 = \emptyset$ we have $k > 0$ and $t\Phi(\prec_{k-1})s$. We follow the cases as distinguished in the definition of Φ. In cases (1) and (2a) we are in the cases (1) and (2a) of this proposition, respectively. In case (2b) we have

$$\langle t_1, \ldots, t_m\rangle \prec_{k-1}^{\tau(f)} \langle s_1, \ldots, s_n\rangle$$

from which we conclude

$$\langle t_1, \ldots, t_m \rangle \prec_{rpo}^{\tau(f)} \langle s_1, \ldots, s_n \rangle$$

by Proposition 44, part (4). Hence we are in case (2b) of the proposition. □

This finishes the proof of Theorem 38.

An alternative definition of recursive path order in which all recursion has been eliminated was given in [41, 42]. However, that definition is not equivalent to ours, and it does not directly imply an effective procedure for deciding whether $s \succ_{rpo} t$ for two given terms s, t.

4.3 Extensions of recursive path order

Many variations and extensions of the basic version of recursive path order have been described in the literature. For an overview we refer to [61]. In this subsection we discuss for all ingredients whether they allow generalizations.

Quasi-orders

One can define a quasi-order $\preceq_{rpo}$ instead of a partial order $\prec_{rpo}$, by a characterization similar to the one given in Theorem 38. In the literature this is the most common way of presenting recursive path order. For termination proofs then the strict part of this quasi-order has to be used, i.e., for every rule $l \to r$ one has to prove that $r \preceq_{rpo} l$ and not $l \preceq_{rpo} r$. Justification of this variant is more complicated than our version; it has been done in detail in [22]. Indeed it is stronger than our version. For instance if f has multiset status, then $f(a, b)$ and $f(b, a)$ are equivalent, and if $f \succ c \succ d$ then in this version one obtains $f(f(a, b), c) \succ_{rpo} f(f(b, a), d)$, which does not hold in our version. In this approach also for the precedence a quasi-order may be chosen, allowing that for distinct but equivalent function symbols f, g clause (2b) is applicable. Another approach to cope with distinct equivalent symbols is to replace them by one single symbol, as is covered by simple preprocessing as we will describe in Subsection 5.1. In the literature many extensions of the quasi-order based recursive path order have been proposed like the *decomposition order*, see for instance [40, 37]. On ground terms they all coincide for total precedences ([59]).

Generalized status

There is no reason to restrict to multiset and lexicographic status as presented in Definition 37. Every way of lifting an order on elements to an order on sequences of elements satisfying the five properties of Proposition 44 will suffice. For instance, if

$$\langle s_1, s_2, s_3 \rangle <^{\tau(f)} \langle t_1, t_2, t_3 \rangle \quad \Leftrightarrow \quad [s_1, s_2] <_{\#} [t_1, t_2] \vee ([s_1, s_2] = [t_1, t_2] \wedge s_3 < t_3),$$

being a combination of lexicographic and multiset status, then

$$f(s(x), y, y) \succ_{rpo} f(y, x, s(x)).$$

This result can neither be obtained by pure multiset status nor by pure lexicographic status.

Semantic path order

Instead of a precedence on function symbols one can choose a precedence being any given quasi-order $\preceq$ on terms, giving the *semantic path order* $\prec_{spo}$ due to [38]. It is characterized by

$$
\begin{array}{lll}
s \succ_{spo} t & \Leftrightarrow & s = f(s_1, \ldots, s_n), \text{ and} \\
& & (1)\ s_i = t \text{ or } s_i \succ_{spo} t \text{ for some } 1 \leq i \leq n, \text{ or} \\
& & (2)\ t = g(t_1, \ldots, t_m),\ s \succ_{spo} t_i \text{ for all } 1 \leq i \leq m, \text{ and either} \\
& & \qquad (a)\ s \succ t, \text{ or} \\
& & \qquad (b)\ s \simeq t \text{ and } \langle s_1, \ldots, s_n \rangle \succ_{spo}^{\tau(f)} \langle t_1, \ldots, t_m \rangle.
\end{array}
$$

where $\succ$ is the reversed strict part of $\preceq$ and $\simeq$ is the equivalence part of $\preceq$. In general the semantic path order is not closed under contexts, and compatibility with $\prec_{spo}$ does not guarantee termination. However, if the quasi-order $\preceq$ satisfies the extra requirement

$$t \to_R u \implies f(\ldots, t, \ldots) \succeq f(\ldots, u, \ldots),$$

then compatibility with $\prec_{spo}$ implies termination, see [38]. This gives a way of proving termination that can also be applied for terminating TRSs that are not simply terminating. The same power of this method is covered by semantic labelling as we shall see in Subsection 5.4.

Recursive path order can be considered as a special case of semantic path order by defining

$$f(t_1, \ldots, t_n) \preceq g(u_1, \ldots, u_m) \quad \Leftrightarrow \quad f \ll g \vee f = g,$$

where $\ll$ is the well-founded precedence on Σ.

Modulo equations

The concept of recursive path order turns out to be hardly applicable for proving termination modulo equations. Only some partial results are available for modulo commutativity (C) and modulo associativity and commutativity (AC). For operation symbols having multiset status the order of the arguments does not play an essential role in the definition of recursive path order: it is easy to see that compatibility with $\prec_{rpo}$ implies termination modulo commutatitivity of the binary symbols having multiset status. Much more complicated are results for termination modulo AC. An important notion with respect to modulo AC is *flattening*: AC-operators are considered as being varyadic operators having an arbitrary finite multiset of arguments. By disallowing direct nesting of these operators, every term has a unique flattened representation. However, some additional requirements have to be fullfilled before termination modulo AC of a TRS can be concluded from rpo-termination of its flattened version, see [7]. Other approaches for variations of recursive path order suitable for proving termination modulo AC are given in [19, 5, 57].

4.4 Knuth-Bendix order

The order we describe here combines the semantical and the syntactical approaches as we discussed them until now. It is a generalization of the original Knuth-Bendix order as described in [43]. The idea of such a generalization goes back to [18].

Definition 57 *A* weakly monotone Σ-algebra $(A, \Sigma_A, <)$ *is a* Σ*-algebra* (A, Σ_A) *provided with a partial order* $<$ *on* A *such that every algebra operation is weakly monotone in all arguments. More precisely, for every* $f \in \Sigma$ *and all* $a_1, \ldots, a_n$, $b_1, \ldots, b_n \in A$ *for which* $a_i < b_i$ *for some* i *and* $a_j = b_j$ *for all* $j \neq i$ *we have* $f_A(a_1, \ldots, a_n) \leq f_A(b_1, \ldots, b_n)$, *where* $\leq$ *is the union of* $<$ *and equality.*

A weakly monotone algebra $(A, \Sigma_A, <)$ *is said to have the* subterm property *if*

$$f_A(a_1, \ldots, a_n) > a_i$$

for every $f \in \Sigma$, $a_1, \ldots, a_n \in A$, $i = 1, \ldots, n$.

Proposition 58 *Let Σ be any signature. Let $(A, \Sigma_A, <)$ be a weakly monotone Σ-algebra, let $\prec$ be a precedence on Σ and let τ be a status on Σ. Then there exists exactly one order $\prec_{kbo}$ on $Ter(\Sigma)$ satisfying*

$$\begin{array}{lll} s \succ_{kbo} t & \Leftrightarrow & s = f(s_1, \ldots, s_n) \text{ and} \\ & & \text{(1) } [\![s]\!]_\alpha > [\![t]\!]_\alpha \text{ for all } \alpha : \mathcal{X} \to A, \text{ or} \\ & & \text{(2) } [\![s]\!]_\alpha \geq [\![t]\!]_\alpha \text{ for all } \alpha : \mathcal{X} \to A, \; t = g(t_1, \ldots, t_m), \text{ and either} \\ & & \qquad \text{(a) } f \succ g, \text{ or} \\ & & \qquad \text{(b) } f = g \text{ and } \langle s_1, \ldots, s_n \rangle \succ_{kbo}^{\tau(f)} \langle t_1, \ldots, t_m \rangle. \end{array}$$

Furthermore, the order $\prec_{kbo}$ is closed under contexts and substitutions.

Proof: Similar to Subsection 4.2 we define $\prec_{kbo}$ to be the least fixed point $\bigcup_{i=0}^{\infty} \Phi^i(\emptyset)$ of the operator Φ on partial orders on $Ter(\Sigma)$ defined by:

$$\begin{array}{lll} t \; \Phi(\sqsubset) \; s & \Leftrightarrow & s = f(s_1, \ldots, s_n) \text{ and} \\ & & (1) \; [\![s]\!]_\alpha > [\![t]\!]_\alpha \text{ for all } \alpha : \mathcal{X} \to A, \text{ or} \\ & & (2) \; [\![s]\!]_\alpha \geq [\![t]\!]_\alpha \text{ for all } \alpha : \mathcal{X} \to A, \; t = g(t_1, \ldots, t_m), \text{ and either} \\ & & \qquad \text{(a) } g \prec f, \text{ or} \\ & & \qquad \text{(b) } g = f \text{ and } \langle t_1, \ldots, t_m \rangle \sqsubset^{\tau(f)} \langle s_1, \ldots, s_n \rangle. \end{array}$$

This definition of Φ is simpler than the version for recursive path order since there is no recursion.

It is straightforward to check that $\Phi(\sqsubset)$ is transitive and irreflexive. Proving uniqueness and closedness under contexts and substitutions, and the verification that the order indeed satisfies the given property are all similar to the corresponding proofs in Subsection 4.2. □

The order $\prec_{kbo}$ as introduced in Propositon 58 is called the *generalized Knuth-Bendix order*. The following proposition states that with some additional requirements it is well-founded, and hence can be applied for termination proofs.

Proposition 59 *Let Σ be any signature. Let $(A, \Sigma_A, <)$ be a weakly monotone Σ-algebra having the subterm property, let $\prec$ be a precedence on Σ and let τ be a status on Σ. If either Σ is finite or both $<$ and $\prec$ are well-founded then the corresponding order $\prec_{kbo}$ is a reduction order.*

Proof: The order $\prec_{kbo}$ is closed under contexts and substitutions by Proposition 58, the only thing to prove is well-foundedness.

The subterm property immediately implies that $\prec_{kbo}$ is a simplification order; if Σ is finite then well-foundedness follows from Proposition 26.

In the remaining case we have that both $<$ and $\prec$ are well-founded. Then it easily follows from the characterization of Proposition 58 that in any infinite sequence that is descending with respect to $\prec_{kbo}$, after a finite initial sequence only clause (2b) is applied. For this case a minimal counterexample argument can be given similar to the proof of Proposition 55. □

In Proposition 59 the subterm property is essential. For instance, if for a unary function symbol f the operation f_A is the identity in some well-founded monotone algebra, then the subterm property does not hold, and we have an infinite descending sequence

$$c \succ_{kbo} f(c) \succ_{kbo} f(f(c)) \succ_{kbo} f(f(f(c))) \succ_{kbo} f(f(f(f(c)))) \cdots$$

for any constant c satisfying $c \succ f$.

A special case of our order is obtained by choosing $(A, <)$ to be the natural numbers $\geq N$ with the usual order, and

$$f_A(k_1, \ldots, k_n) = w(f) + k_1 + k_2 + \cdots + k_n$$

for all $f \in \Sigma$, and choosing τ to be a lexicographic status. This special case corresponds to the original Knuth-Bendix order. There for every $f \in \Sigma \cup \mathcal{X}$ a weight $w(f)$ is defined, while $w(x) = N$ for all $x \in \mathcal{X}$ for some positive constant $N \in \mathbb{N}$. Similar to the proof of Lemma 23 one easily verifies

$$\forall \alpha \ : \ [\![s]\!]_\alpha > [\![t]\!]_\alpha \quad \Leftrightarrow \quad Var(t) \subseteq_\# Var(s) \wedge W(s) > W(t),$$

where $W(u)$ is defined to be the total weight of a term u, and $Var(u)$ denotes the multiset of variables in u. For the algebra to be well-defined we need the requirement that $w(c) \geq N$ for constants c; for the subterm property we need the requirement that $w(f) > 0$ for unary symbols f. These are exactly the requirements as they appear in the original Knuth-Bendix order. Note that these are quite restrictive, for instance if a TRS contains a duplicating rule $l \to r$ then $l \succ_{kbo} r$ never holds.

Example 18 Termination of the TRS consisting of the two rules

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

can not be proven by recursive path order. However, by Knuth-Bendix order this is easy by choosing A to consist of the natural numbers, $f_A(x) = g_A(x) = x + 1$, $f \succ g$. In the original terminology of weights, one has $w(f) = w(g) = 1$.

In the version of [20] it is allowed that $w(f) = 0$ for one unary symbol f if $f \succ g$ for all other symbols g; it is easily checked that for this version the subterm property is also implied.

For using $\prec_{kbo}$ as a reduction order for mechanizing termination proofs, one needs a procedure to find a suitable weakly monotone algebra and suitable $\succ, \tau$ such that $l \succ_{kbo} r$ for all rewrite rules $l \to r$. For the restricted version described above such a procedure has been given in [20].

5 Transformational methods

Until now we described some basic methods to prove termination of a given TRS. For many terminating TRSs these methods still fail to prove termination. In the literature many attempts have been made for strengthening these methods, mainly by refining path orders. We prefer another approach: if basic orders like recursive path order fail to prove termination of a TRS R, we will not try to find refinings of the order, but we try to apply a non-termination preserving transformation Ψ on R such that termination of $\Psi(R)$ can be proved by means of recursive path order. Here *non-termination preserving* is defined as follows.

Definition 60 *Let Ψ be a transformation giving for a TRS (Σ, R) a new TRS $\Psi(\Sigma, R) = (\Psi(\Sigma), \Psi(R))$. Then Ψ is called* non-termination preserving *if termination of (Σ, R) follows from termination of $\Psi(\Sigma, R)$.*

By definition termination of R has been proved as soon as a non-termination preserving transformation Ψ can be found for which termination of $\Psi(R)$ can be proved, e.g. by recursive path order. In this way developing non-termination preserving transformations on TRSs gives rise to a new class of transformational methods to prove termination of TRSs. These methods may freely be combined: if termination of $\Psi(R)$ cannot be proved by basic methods, we can go on by trying to

find another transformation Ψ' for which termination of $\Psi'(\Psi(R))$ is easily proved. If this succeeds for non-termination preserving transformations Ψ, Ψ', then indeed we may conclude termination of R, and so on.

In this section we give an overview of non-termination preserving transformations.

5.1 Basic transformations

Consider the TRS consisting of the two rules

$$\begin{array}{rcl} g(f(x)) & \to & f(h(x)) \\ h(x) & \to & g(x). \end{array}$$

Termination is easily proved by the polynomials $f_A = X+1, g_A = 3X, h_A = 3X+1$. Surprisingly, termination of this TRS can not be proved by recursive path order or Knuth-Bendix order. Seeing what is going on this is quite strange: the second rule is nothing else than a renaming of the symbol h to g, and applying this renaming to the first rule gives $g(f(x)) \to f(g(x))$ of which termination is easily proved by recursive path order by choosing the precedence $g \succ f$. In this subsection we generalize this idea of renaming to a non-termination preserving transformation. The result can be considered as a syntactical version of Proposition 14.

In a renaming of symbols every occurrence of a symbol is replaced by another symbol. We generalize this to recursive program schemes. In Section 5.5 we shall see that they also play an important role in the dependency pair approach to proving termination.

Definition 61 *A* recursive program scheme (RPS) *is a TRS in which all left hand sides of the rules have distinct root symbols, and all of these left hand sides are of the shape $f(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are distinct variables.*

Every RPS is confluent since it is orthogonal.

For a finite RPS consider the directed graph of which the nodes are the operation symbols, and there is an edge from f to g if and only there is a rule $l \to r$ in the RPS for which f occurs in l and g occurs in r. It is not difficult to see that the RPS is terminating if and only if this directed graph is acyclic. Hence termination of finite RPSs is easy to establish. In particular, a finite RPS is terminating if and only if it is rpo-terminating.

Definition 62 *Let S be a terminating RPS. For a term t we write $S(t)$ for the unique normal form of t with respect to S. For a substitution σ we write $S(\sigma)$ for the substitution defined by $x^{S(\sigma)} = S(x^\sigma)$ for all variables x.*

Before we state and prove our theorem, we give two lemmas.

Lemma 63 *Let S be a terminating RPS, let t be a term and let σ be a substitution. Then*

$$S(t^\sigma) = S(t)^{S(\sigma)}.$$

Proof: Clearly $t^\sigma \to_S^* S(t^\sigma)$ and $t^\sigma \to_S^* S(t)^\sigma \to_S^* S(t)^{S(\sigma)}$. Since S is confluent and $S(t)^\sigma$ is a normal form with respect to S, it remains to show that $S(t)^{S(\sigma)}$ is a normal form with respect to S too. Since $S(t)$ is a normal form it does not contain symbols occurring in left hand sides of S. Since $x^{S(\sigma)} = S(x^\sigma)$ is a normal form it does not contain symbols occurring in left hand sides of S for all variables x in $S(t)$. Hence $S(t)^{S(\sigma)}$ does not contain symbols occurring in left hand sides of S, hence it is a normal form with respect to S. □

Lemma 64 *Let S be a terminating non-erasing RPS, let R be any TRS and let t, u be terms satisfying $S(t) \to_R^+ S(u)$. Then*

$$S(C[t]) \to_R^+ S(C[u])$$

for every context C.

Proof: Applying induction on the structure of C it suffices to prove that

$$S(f(t_1, \ldots, t_i, t, t_{i+1}, \ldots, t_n)) \to_R^+ S(f(t_1, \ldots, t_i, u, t_{i+1}, \ldots, t_n)),$$

where we assume that $S(t) \to_R^+ S(u)$. Write $v = f(x_1, \ldots, x_n)$ for n arbitrary distinct variables $x_1, \ldots, x_n$. Define σ and τ by $x_j^\sigma = x_j^\tau = t_j$ for all $j \neq i$, and $x_i^\sigma = t$ and $x_i^\tau = u$. Since S is non-erasing, the variable x_i occurs at least once in $S(v)$. Since $x_i^{S(\sigma)} = S(x_i^\sigma) = S(t) \to_R^+ S(u) = S(x_i^\tau) = x_i^{S(\tau)}$, and $x_j^{S(\sigma)} = S(x_j^\sigma) = S(t_j) = S(x_j^\tau) = x_j^{S(\tau)}$ for all $j \neq i$, by applying Lemma 63 we obtain

$$\begin{array}{lcl} S(f(t_1, \ldots, t_i, t, t_{i+1}, \ldots, t_n)) & = & S(v^\sigma) \\ & = & S(v)^{S(\sigma)} \\ & \to_R^+ & S(v)^{S(\tau)} \\ & = & S(v^\tau) \\ & = & S(f(t_1, \ldots, t_i, u, t_{i+1}, \ldots, t_n)). \end{array}$$

□

Theorem 65 *Let R be any TRS. Let S be a terminating non-erasing RPS and let*

$$R_1 = \{l \to r \in R \mid S(l) = S(r)\},$$

$$R_2 = \{S(l) \to S(r) \mid l \to r \in R \wedge S(l) \neq S(r)\}.$$

If both R_1 and R_2 are terminating TRSs, then R is terminating.

Proof: Let $R_3 = \{l \to r \in R \mid S(l) \neq S(r)\}$. Then R is the disjoint union of R_1 and R_3. Assume R admits an infinite reduction

$$t_1 \to_R t_2 \to_R t_3 \to_R t_4 \to_R \cdots.$$

For every step $t_i \to_R t_{i+1}$ we have $t_i = C[l^\sigma]$ and $t_{i+1} = C[r^\sigma]$ for some context C, some substitution σ, and some rule $l \to r$ in either R_1 or R_3. If $l \to r$ is a rule in R_1 then we have

$$S(t_i) = S(C[l^\sigma]) = S(C[S(l)^\sigma]) = S(C[S(r)^\sigma]) = S(C[r^\sigma]) = S(t_{i+1}).$$

In the remaining case $l \to r$ is a rule in R_3. Then $S(l) \to S(r)$ is a rule in R_2. Applying Lemma 63 yields

$$S(l^\sigma) = S(l)^{S(\sigma)} \to_{R_2} S(r)^{S(\sigma)} = S(r^\sigma).$$

Then applying Lemma 64 yields

$$S(t_i) = S(C[l^\sigma]) \to_{R_2}^+ S(C[r^\sigma]) = S(t_{i+1}).$$

We conclude that $S(t_i) \to_{R_2}^* S(t_{i+1})$ for all i, and $S(t_i) \to_{R_2}^+ S(t_{i+1})$ if $t_i \to_{R_3} t_{i+1}$. Since R_1 is terminating the original infinite R-reduction contains infinitely many R_3-steps. Hence this latter case occurs infinitely often, yielding an infinite R_2-reduction, contradiction. □

Example 19 Let the TRS R consist of the two rules

$$\begin{array}{rcl} g(f(x)) & \to & f(h(x)) \\ h(x) & \to & g(x). \end{array}$$

We started this subsection by the remark that termination of R can neither be proved by recursive path order nor by Knuth-Bendix order. However, by choosing S to consist of the single rule $h(x) \to g(x)$, we obtain

$$R_1 = \{h(x) \to g(x)\},$$

$$R_2 = \{g(f(x)) \to f(g(x))\},$$

which are both proved to be terminating by means of recursive path order. By Theorem 65 termination of R follows.

Example 20 Let the TRS R consist of the three rules

$$\begin{array}{rcl} f(x) & \to & g(x) \\ g(f(x)) & \to & h(x) \\ g(h(x)) & \to & f(x). \end{array}$$

By recursive path order as we presented it, termination of R can not be proved. By versions described in the literature based on precedences that are quasi-orders instead of partial orders it easily can by choosing f and h to be equivalent, both being greater than g. However, taking symbols to be equivalent is easily simulated using Theorem 65 by choosing one representant in an equivalence class and by choosing S to be the rewriting from the other equivalent symbols to this representant. In this case where f and h are equivalent we may choose f as the representant, yielding S to consist of the rule $h(x) \to f(x)$. In this example then R_1 is empty and R_2 consists of the rules

$$\begin{array}{rcl} f(x) & \to & g(x) \\ g(f(x)) & \to & f(x), \end{array}$$

easily proved to be terminating by our basic version of recursive path order. By Theorem 65 termination of R follows.

Theorem 65 yields a non-termination preserving transformation: for any TRS R we may define $\Psi(R) = R_2$ in the notation of Theorem 65, for any non-erasing RPS S for which the corresponding TRS R_1 is terminating.

One can wonder whether the requirements on S in Theorem 65 are really essential. Indeed they are. For the normal form $S(t)$ being well-defined for every term t the TRS S needs to be confluent and weakly normalizing. If we choose R to consist of the rule $f(x) \to f(f(x))$ and S to consist of the rule $f(f(x)) \to x$ then S is terminating and confluent, R_1 is empty and $R_2 = \{f(x) \to x\}$ is terminating, but R itself is not terminating. Hence the requirement of S being an RPS cannot be weakened to S being confluent. Also non-erasingness is essential: if we choose R to consist of the rule $g(x) \to f(g(x))$ and S to consist of the rule $f(x) \to a$ then S is a terminating RPS, R_1 is empty and $R_2 = \{g(x) \to a\}$ is terminating, but R itself is not terminating. Finally we remark that every RPS is orthogonal by definition, and it is well-known that termination and weak normalization coincide for non-erasing orthogonal TRSs.

5.2 Dummy elimination

In this section we show how the task of proving termination can be simplified fully automatically in case some symbol does not occur in any left hand side of a rule.

Such a symbol is called a *dummy symbol*, or shortly *dummy*. For instance, in the rewrite system R consisting of the single rule

$$f(g(x)) \;\to\; f(a(g(x)))$$

the symbol a does not occur in a left hand side. Intuitively this means that this dummy symbol does not play an essential role in further reductions of the term, and further reductions can be localized as being either affecting the part above the dummy symbol or affecting the part below the dummy symbol. This can be formalized by decomposing the right hand sides into smaller terms in which the dummy acts as a separator. In this case this means that the term $f(h(g(x)))$ is decomposed into two terms $f(\diamond)$ and $g(x)$, where $\diamond$ is a fresh constant. The left hand sides remain the same. The result is the transformed system $E(R)$, in our case consisting of the two rules

$$\begin{array}{lcl} f(g(x)) & \to & f(\diamond) \\ f(g(x)) & \to & g(x). \end{array}$$

The main theorem states that this dummy elimination E is a non-termination-preserving transformation, i.e., termination of R can be proved by proving termination of $E(R)$. In our example this is valuable: R is not simply terminating. On the other hand $E(R)$ is simply terminating, which is trivially seen since every $E(R)$-step strictly decreases the term size.

In order to give a precise definition for dummy elimination we need some auxiliary definitions. We fix a dummy symbol a, i.e., a symbol not occurring in the left hand side of any rule in a fixed TRS R. Let n be the arity of a. For any term t we define inductively a term $\mathsf{cap}(t)$ and a set of terms $\mathsf{dec}(t)$:

$$\begin{array}{lcll} \mathsf{cap}(x) & = & x & \text{for all } x \in \mathcal{X}, \\ \mathsf{cap}(f(t_1,\ldots,t_k)) & = & f(\mathsf{cap}(t_1),\ldots,\mathsf{cap}(t_k)) & \text{for all } f \text{ with } f \neq a \\ \mathsf{cap}(a(t_1,\ldots,t_n)) & = & \diamond & \\ \mathsf{dec}(x) & = & \emptyset & \text{for all } x \in \mathcal{X}, \\ \mathsf{dec}(f(t_1,\ldots,t_k)) & = & \bigcup_{i=1}^{k} \mathsf{dec}(t_i) & \text{for all } f \text{ with } f \neq a \\ \mathsf{dec}(a(t_1,\ldots,t_n)) & = & \bigcup_{i=1}^{n} (\mathsf{dec}(t_i) \cup \{\mathsf{cap}(t_i)\}). & \end{array}$$

This means that if a term t is decomposed by considering the symbol a as a separator, then the term $\mathsf{cap}(t)$ is the rootmost part of this decomposition, while $\mathsf{dec}(t)$ is the set of all other parts in this decomposition. Now we define the dummy elimination construction and state the main theorem.

Definition 66 *Let a be a dummy symbol in a TRS R and let* cap *and* dec *be defined as above. Then*

$$E(R) = \{l \to u \mid u = \mathsf{cap}(r) \vee u \in \mathsf{dec}(r) \text{ for a rule } l \to r \in R\}.$$

Theorem 67 *Let a be a dummy symbol in a TRS R. If $E(R)$ is terminating then R is terminating too.*

For a proof of this theorem we refer to [25] or [22], where a slightly more general version has been treated. An alternative proof has been given in [51], where even the restriction of the dummy not occurring in left hand sides has been weakened slightly. A generalization of this result to rewriting modulo equations has been given in [23].

Example 21 For proving termination of the TRS R consisting of the two rules

$$\begin{array}{lcl} f(g(x)) & \to & f(a(g(g(f(x))), g(f(x)))) \\ g(f(x)) & \to & g(g(a(f(x), g(g(x))))) \end{array}$$

it suffices by Theorem 67 to prove termination of the TRS $E(R)$ consisting of the rules

$$\begin{array}{lcl} f(g(x)) & \to & f(\diamond) \\ f(g(x)) & \to & g(g(f(x))) \\ f(g(x)) & \to & g(f(x)) \\ g(f(x)) & \to & g(g(\diamond)) \\ g(f(x)) & \to & f(x) \\ g(f(x)) & \to & g(g(x)) \end{array}$$

which is easily done by recursive path order, choosing the precedence $f > g > \diamond$.

Exercise 10 Prove termination of the TRS R consisting of the single rule

$$f(g(x,y)) \to g(f(h(g(x,f(y)),f(y))),g(y,f(x))).$$

We conclude this section by a variant of dummy elimination from [64] in which also rules describing distributivity are allowed, that's why this variant is called *distribution elimination*. A rewrite rule is called a *distribution rule* for a if it can be written as

$$C[a(x_1,\ldots,x_n)] \to a(C[x_1],\ldots,C[x_n])$$

for some non-trivial context $C[\,]$ in which the symbol a does not occur. For example,

$$b(z,f(a(x,y)) \to a(b(z,f(x)),b(z,f(y)))$$

is a distribution rule for a. Now E_a is defined inductively as follows:

$$\begin{array}{lcll} E_a(x) & = & \{x\} & \text{for all } x \in \mathcal{X}, \\ E_a(f(t_1,\ldots,t_k)) & = & \{f(u_1,\ldots,u_k) \mid \forall i : u_i \in E_a(t_i)\} & \text{for all } f \text{ with } f \neq a \\ E_a(a(t_1,\ldots,t_n)) & = & \bigcup_{i=1}^{n} E_a(t_i). & \end{array}$$

Let R be a TRS for which each rule is either a distribution rule for a or a rule in which a does not occur in the left hand side. Then the TRS $E_a(R)$ is defined by

$$E_a(R) = \{l \to u \mid l \to r \text{ is a non-distribution rule of } R \text{ for } a \text{ and } u \in E_a(r)\}.$$

As usual a term is defined to be linear if no variable occurs more than once, and a TRS is defined to be right-linear if for every rule the right hand side is linear.

Theorem 68 *Let R be a TRS for which each rule is either a distribution rule for the symbol a or a rule in which the symbol a does not occur in the left hand side. Then*

(1) *$E_a(R)$ is totally terminating if and only if R is totally terminating;*
(2) *if $E_a(R)$ is right-linear, then*
$E_a(R)$ is simply terminating if and only if R is simply terminating;
(3) *if $E_a(R)$ is terminating and right-linear then R is terminating.*

For the proof we refer to [64]. There also examples are given showing that the converse of (3) does not hold and showing that the requirement of right-linearity is essential in both (2) and (3). It was conjectured that in absense of distribution rules the restriction of right-linearity is not essential in (3); this conjecture was proved in [51]. In [50] Theorem 68 has been used to prove undecidability of simple termination of single rewrite rules.

5.3 Applying abstract commutation

In this section we describe a few ways of how to prove termination of a TRS R by means of proving termination of some modified TRS $S = \Psi(R)$. This can be proved to be valid if an auxiliary TRS T essentially describing the difference between R and S, satisfies some commutation properties. The underlying framework is described purely in terms of abstract reduction systems, that's why we coin these methods *abstract commutation*.

In the following R, S and T are arbitrary binary relations on a fixed set. In the applications they will correspond to rewrite relations of TRSs. As usual we write a dot symbol for relational composition, i.e., one has $t(R \cdot S)t'$ if and only if there exists a t'' such that tRt'' and $t''St'$. We write R^+ for the transitive closure of R and R^* for the reflexive transitive closure of R, and we write R^{-1} for the inverse of R. Further we write $R \subseteq S$ if tRt' implies tSt'. Clearly, if $R \subseteq S$ then $R \cdot T \subseteq S \cdot T$ and $T \cdot R \subseteq T \cdot S$. In the following lemma we collect some standard properties for relations, which are easy to check.

Lemma 69 *Let R, S, T be binary relations.*
(a) If $R \cdot S \subseteq S \cdot R^*$, then $R^* \cdot S \subseteq S \cdot R^*$.
(b) If $T^* \cdot R \subseteq S^+ \cdot T^*$ and S is terminating, then R is terminating.

The first abstract commutation theorem we present is the version from [68]; a direct generalization of this theorem to rewriting modulo equation has been given in [28].

Theorem 70 *Let R, S, T be binary relations satisfying*

1. *S is terminating,*
2. *$R \subseteq S^+ \cdot (T^{-1})^*$,*
3. *$T^{-1} \cdot R \subseteq R^+ \cdot (T^{-1})^*$,*

then R is terminating.

Before giving the proof of this theorem we provide some intuition. Suppose that R is not terminating, so there is an infinite reduction $R \cdot R \cdot R \cdots$. Using condition 2, the leftmost R-step in this reduction can be replaced by $S^+ \cdot (T^{-1})^*$. The created T^{-1}-steps jump over R-steps by applying condition 3. Then the infinite reduction starts with $S^+ \cdot R$, and the whole process can be applied on the leftmost R-step in this reduction again. Repeating this construction forever yields an infinite S-reduction, contradicting condition 1.
Proof: Using condition 3 we obtain:

$$T^{-1} \cdot R \subseteq R^+ \cdot (T^{-1})^* \subseteq R^+ \cdot (R \cup T^{-1})^* = R \cdot (R \cup T^{-1})^*.$$

Since also $R \cdot R \subseteq R \cdot (R \cup T^{-1})^*$, we obtain $(R \cup T^{-1}) \cdot R \subseteq R \cdot (R \cup T^{-1})^*$. From Lemma 69, part (a), and condition 2 we conclude

$$(R \cup T^{-1})^* \cdot R \subseteq R \cdot (R \cup T^{-1})^* \subseteq S^+ \cdot T^{-1})^* \cdot (R \cup T^{-1})^* = S^+ \cdot (R \cup T^{-1})^*.$$

Now termination of R follows from condition 1 and Lemma 69, part (b). □

In Theorem 70 condition 3 is the commutation criterion, and condition 2 relates R and S. In the latter it is required that every R-step can be followed by zero or more T-steps to obtain one or more S-steps.

Before describing applications in term rewriting we first present another abstract commutation theorem, from [10], based on [9, 6]. Here the requirement in condition

2 on the order of S-steps and T-steps is weakened; the commutation criterion between R and T is replaced by a commutation criterion between S and T, and also confluence and termination are needed for T. As in [29], we write S/T for $T^* \cdot S \cdot T^*$.

Theorem 71 *Let R, S, T be binary relations satisfying*

1. *$S \cup T$ is terminating,*
2. *$R \subseteq (S/(T \cup T^{-1}))^+$,*
3. *$T^{-1} \cdot S \subseteq T^* \cdot S \cdot (S \cup T \cup T^{-1})^*$,*
4. *$T^{-1} \cdot T \subseteq T^* \cdot (T^{-1})^*$,*

then R is terminating.

Here condition 2 requires that every R-step can be simulated by any combination of S-steps, T-steps and T^{-1}-steps with the only restriction that it contains at least one S-step. Roughly speaking we can say that the theorem states that in an infinite reduction of this shape all T^{-1}-steps can be infinitely thrown forwardly by using the other conditions. In the first approach into this direction ([6]) the condition corresponding to condition 2 was $R \subseteq T^* \cdot S \cdot (T^{-1})^*$, being some stronger. Condition 4 is local confluence of T; since T is terminating by condition 1 by Newman's lemma this is equivalent to confluence of T. Condition 3 is called *local cooperation*. Our version of local cooperation (suggested by Vincent van Oostrom) is slightly weaker than the original version $T^{-1} \cdot S \subseteq (S/T)^+ \cdot (T^{-1})^*$ from [10], making the theorem slightly stronger.

Another way of stating the same theorem without referring to R and the second condition is to conclude that $(S/(T \cup T^{-1}))^+$ is a well-founded order. In [9] this order is called the *transformation ordering*; compatibility with this order then is usually shown by proving $T(l) \to_S T(r)$ for all $l \to r$ in R, where $T(-)$ denotes reducing to normal form with respect to T.

In order to prove Theorem 71 we need the following lemma.

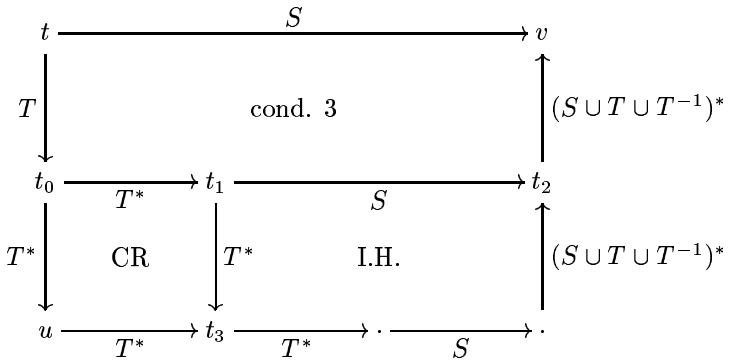
Lemma 72 *Let S and T binary relations satisfying conditions 1, 3 and 4 from Theorem 71. Then*

$$(T^{-1})^* \cdot S \subseteq T^* \cdot S \cdot (S \cup T \cup T^{-1})^*.$$

Proof: We prove this lemma by proving

$$\text{If } tT^*u \text{ and } tSv \text{ for any } u, v, \text{ then } uT^* \cdot S \cdot (S \cup T \cup T^{-1})^* v \text{ holds}$$

by well-founded induction on t over the well-founded order T^+. If $t = u$ then this is trivial, hence we may assume that tT^+u. Then some t_0 exists satisfying tTt_0 and t_0T^*u. From condition 3 from Theorem 71 we conclude that there exists elements t_1 and t_2 satisfying $t_0T^*t_1$, t_1St_2 and $t_2(S \cup T \cup T^{-1})^*v$. From conditions 1 and 4 from Theorem 71 and Newman's Lemma we conclude that T is confluent, hence t_3 exists satisfying uT^*t_3 and $t_1T^*t_3$. Since tT^+t_1 we may apply the induction hypothesis on t_1. Since $t_1T^*t_3$ and t_1St_2 this yields $t_3T^* \cdot S \cdot (S \cup T \cup T^{-1})^*t_2$. Combined with uT^*t_3 and $t_2(S \cup T \cup T^{-1})^*v$ we obtain $uT^* \cdot S \cdot (S \cup T \cup T^{-1})^*v$, which we had to prove. In a diagram this proof reads:



□

Now we prove Theorem 71.

Proof: Since T is confluent we obtain $(T \cup T^{-1})^* \subseteq T^* \cdot (T^{-1})^*$. Applying this, condition 2 and Lemma 72 yields

$$\begin{array}{lll} (S \cup T \cup T^{-1})^* \cdot R & \subseteq & (S \cup T \cup T^{-1})^* \cdot (S/(T \cup T^{-1}))^+ \\ & = & (T \cup T^{-1})^* \cdot S \cdot (S \cup T \cup T^{-1})^* \\ & \subseteq & T^* \cdot (T^{-1})^* \cdot S \cdot (S \cup T \cup T^{-1})^* \\ & \subseteq & T^* \cdot S \cdot (S \cup T \cup T^{-1})^* \\ & \subseteq & (S \cup T)^+ \cdot (S \cup T \cup T^{-1})^*. \end{array}$$

Now from condition 1 and Lemma 69, part (b), we conclude termination of R. This concludes the proof of Theorem 71. □

We want to apply Theorems 70 and 71 for the case where $R = \rightarrow_R$ for some TRS R for which termination has to be proved, $S = \rightarrow_S$ for some TRS S which is some modification of R, and $T = \rightarrow_T$ for some TRS T chosen in such a way that the requirements of the corresponding theorem hold. As usual the local confluence criterion, condition 4 of Theorem 71, is verified by checking critical pairs. But also condition 3 of Theorem 70 and condition 3 of Theorem 71 can be proved using the same idea of critical pairs: a finite case analysis has to be done according to all possiblities of overlap, and the non-overlapping case is obtained for free if some extra conditions hold. To make this precise first we define the corresponding notion of critical pairs.

Definition 73 *A* critical pair *between two rewrite rules $l \rightarrow r$ and $l' \rightarrow r'$ is a pair of terms of one of the two following shapes:*

- *$(r^\sigma, C^\sigma[r'^\sigma])$, where $l = C[t]$ for some non-variable term t, and σ is the most general unifier of t and l',*
- *$(C^\sigma[r^\sigma], r'^\sigma)$, where $l' = C[t]$ for some non-variable term t, and σ is the most general unifier of t and l.*

Here variables may be renamed in such a way that l and l' have no variables in common.

For two TRSs R and S the full set of critical pairs between any rule of R and any rule of S is denoted by $CP(R, S)$.

Note that for any TRS R the set $CP(R, R)$ consists of the pairs (r, r) for all rules $l \rightarrow r$ in R and all pairs (s, t) and (t, s) where (s, t) is a critical pair of R in the usual definition.

Note that for finite TRSs R and S the set $CP(R,S)$ is finite two, and is easily computed by unifying left hand sides of R with subterms of left hand sides of S and vice versa. The following theorem shows how condition 3 of Theorem 70 and condition 3 of Theorem 71 can effectively be verified.

Theorem 74 *Let T and A be TRSs of which T is left-linear and non-erasing. Let C be a rewrite relation satisfying $\rightarrow_T^* \cdot \rightarrow_A^+ \cdot \leftarrow_T^* \subseteq C$; if A is left-linear then this restriction on C may be weakened to $\rightarrow_A^+ \cdot \leftarrow_T^* \subseteq C$. Then $\leftarrow_T \cdot \rightarrow_A \subseteq C$ if and only if $CP(T,A) \subseteq C$.*

The proof is not given here since it is similar to the proof of the well-known critical pair lemma. It generalizes Theorem 2 from [10].

In the rest of this section we restate Theorems 70 and 71 specified for TRSs using this result on critical pairs, and give a number of applications.

Theorem 75 *Let R, S, T be TRSs satisfying*

1. *S is terminating,*
2. *$R \subseteq \rightarrow_S^+ \cdot \leftarrow_T^*$,*
3. *$CP(T,R) \subseteq \rightarrow_R^+ \cdot \leftarrow_T^*$.*
4. *T is left-linear and non-erasing,*
5. *R is left-linear,*

then R is terminating.

Note that condition 2 is only about rewrite rules of R and not about rewrite steps. Hence for finite TRSs conditions 2 to 5 can be verified by a finite analysis.
Proof: Condition 1 corresponds to condition 1 of Theorem 70. Since $\rightarrow_S^+ \cdot \leftarrow_T^*$ is closed under contexts and substitutions, from condition 2 we conclude $\rightarrow_R \subseteq \rightarrow_S^+ \cdot \leftarrow_T^*$, corresponding to condition 2 of Theorem 70. Finally condition 3 of Theorem 70 holds by conditions 3, 4 and 5 and Theorem 74, choose $A = R$ and $C = \leftarrow_T \cdot \rightarrow_A$. Hence by Theorem 70 we may conclude termination of $\rightarrow_R$. □

Example 22 Let R consist of the single rule $f(g(x)) \rightarrow g(f(h(g(x))))$. Clearly R is not simply terminating. By choosing S to consist of the single rule $f(g(x)) \rightarrow g(f(x))$ and T to consist of the single rule $h(g(x)) \rightarrow x$, then clearly the conditions 2 to 5 hold, note that $CP(T,R) = \emptyset$. Hence for proving termination of R by Theorem 75 it suffices to prove termination of S, which can be done by recursive path order.

This example could also be treated by dummy elimination, which does not hold for the following example.

Example 23 Let R consist of the single rule $f(h(g(x))) \rightarrow g(f(h(f(g(x)))))$. Clearly R is not simply terminating and does not contain dummy symbols. By choosing S to consist of the single rule $f(h(g(x))) \rightarrow g(f(x))$ and T to consist of the single rule $h(f(g(x))) \rightarrow x$, then clearly the conditions 2 to 5 hold, again $CP(T,R) = \emptyset$. Hence for proving termination of R by Theorem 75 it suffices to prove termination of S, which can be done by counting h-symbols.

The intuition behind these two examples is that the problematic right hand side, in these cases the right hand side of the single rule, contains a "dead part": a part that does not overlap with any left hand side. Such a dead part cannot play an essential role in future reductions. In these examples S is obtained from R by

cutting away the dead part, and T is a rewrite system that executes this process of cutting away the dead part.

Sometimes it is convenient not to remove the full dead part, but to replace it by a term containing a fresh symbol. In this way this fresh symbol will occur in one or more right hand sides of S, but not in left hand sides. Hence the fresh symbol is a dummy in S, and we can apply dummy elimination to prove termination of S. This way of introducing a dummy symbol is called *dummy introduction* in [68].

Example 24 Let R consist of the single rule $f(g(f(x))) \to g(f(f(g(g(f(f(x)))))))$. Here the two consecutive g symbols in the right hand side do not overlap with the left hand side, hence act as a dead part. By choosing T to consist of the rule $g(g(x)) \to h(x)$, for S we obtain the single rule $f(g(f(x))) \to g(f(f(h(f(f(x))))))$. Since $CP(T,R) = \emptyset$ the condition 2 to 5 are all satisfied and it remains to show termination of S. Due to Theorem 67 where h is the dummy symbol, it remains to show termination of $E(S)$, consisting of the two rules

$$\begin{array}{lcl} f(g(f(x))) & \to & g(f(f(\diamond))) \\ f(g(f(x))) & \to & f(f(x)), \end{array}$$

which is done by recursive path order choosing $f > g > \diamond$.

Exercise 11 Prove termination of $f(g(f(x))) \to g(g(f(f(g(x)))))$.

In the next example we do not have an empty set of critical pairs.

Example 25 As in Example 7 let R be the simplest terminating TRS that is not simply terminating: the single rule $f(f(x)) \to f(g(f(x)))$. Choose S to consist of the single rule $f(f(x)) \to f(x)$ and T to consist of the single rule $f(g(f(x))) \to f(x)$. Then clearly conditions 2, 4 and 5 of Theorem 75 hold. We obtain $CP(T,R) = \{(f(f(x)), f(g(f(g(f(x))))))\}$, note that this single critical pair is obtained in two ways. Since $f(f(x)) \to_R f(g(f(x)))$ and $f(g(f(g(f(x))))) \to_T f(g(f(x)))$, also condition 3 holds. Since S is terminating, by Theorem 75 is terminating too.

Exercise 12 Prove termination of $f(f(g(x))) \to g(f(g(f(g(x)))))$. (Hint: choose $T : g(f(g(x))) \to g(x)$.)

Often it occurs that the non-empty set of critical pairs does not satisfy the required condition 3. One way to solve this is to extend the TRS T by some extra rules forced by this condition. Note that if only non-erasing left-linear rules are added, the other four conditions are not affected by adding these rules to T. Similar to Knuth-Bendix completion after adding extra rules new critical pairs have to be computed and the process goes on. This is one of the ingredients of *termination by completion* as it is described in [10]. Just like we know in Knuth-Bendix completion that some completions will never succeed since the corresponding word problem is not solvable, here we know that some completions wil never succeed since the system is not terminating. For instance, if R consists of the rule $f(x) \to f(g(x))$ and we try to start with S consisting of the rule $f(x) \to x$ and T consisting of the rule $f(g(x)) \to x$, we know for sure that we can never extend T in such a way that condition 3 holds, since R is not terminating. In [68] an example of a successfull completion of this kind is given. The same holds for the result in [28], there even the systems R and S are extended during the completion process, and the final system T has infinitely many rules. Here we only give a simple example illustrating the idea.

Example 26 Let R consist of the two rules

$$\begin{array}{lcl} f(g(x)) & \to & f(h(g(x))) \\ h(g(x)) & \to & f(x). \end{array}$$

As a first attempt we choose T to consist of the single rule $f(h(x)) \to h(x)$, and for S we obtain the two rules

$$\begin{array}{lcl} f(g(x)) & \to & h(g(x)) \\ h(g(x)) & \to & f(x), \end{array}$$

of which termination is easily verified by recursive path order, choosing $g > f > h$. However, we do not yet have condition 3:

$$CP(T,R) = \{(h(g(x)), f(f(x)))\} \not\subseteq \to_R^+ \cdot \leftarrow_T^* .$$

It is possible to rewrite $h(g(x)) \to_R^+ f(x)$, but it is not yet possible to rewrite $f(f(x)) \to_T^* f(x)$. This is simply solved by adding the rule $f(f(x)) \to f(x)$ to T. Now the old critical pair satisfies the requirement, and we have to go on for checking whether there are new critical pairs. Indeed there are: the left hand side of this new rule overlaps with the left hand side of the first rule of R, giving the new critical pair $(f(g(x)), f(f(h(g(x)))))$. Since $(f(g(x)) \to_R^+ f(h(g(x))) \leftarrow_T^* f(f(h(g(x))))$, indeed now

$$CP(T,R) = \{(h(g(x)), f(f(x))), (f(g(x)), f(f(h(g(x)))))\} \subseteq \to_R^+ \cdot \leftarrow_T^*,$$

and all conditions of Theorem 75 hold, proving termination of R.

Next we show that in Theorem 75 left-linearity for both R and T and non-erasingness of T are all essential.

If $R = \{f(x,x) \to f(g(x), g(x))\}$, $T = \{g(x) \to x\}$ and $S = \{f(x,x) \to f(x, g(x))\}$, then all conditions of Theorem 75 hold except for left-linearity of R, while R is not terminating.

If $R = \{f(a,b) \to f(a,a), a \to b\}$, $T = \{f(x,x) \to x\}$ and $S = \{f(a,b) \to a, a \to b\}$, then all conditions of Theorem 75 hold except for left-linearity of T, while R is not terminating.

If $R = \{a \to f(a)\}$, $T = \{f(x) \to b\}$ and $S = \{a \to b\}$, then all conditions of Theorem 75 hold except for non-erasingness of T, while R is not terminating.

Until now all examples used Theorem 75, based upon the abstract commutation theorem Theorem 70. The following theorem makes the abstract commutation theorem Theorem 71 applicable to term rewriting, by which the requirement that R-steps are simulated by first S-steps and then inverse T-steps, is essentially weakened. Moreover, it does not restrict any more to left-linear TRSs.

Theorem 76 *Let R, S, T be TRSs satisfying*

1. *$S \cup T$ is terminating,*
2. *$R \subseteq (\to_S / (\to_T \cup \leftarrow_T))^+$,*
3. *$CP(T,S) \subseteq \to_T^* \cdot \to_S \cdot (\to_S \cup \to_T \cup \leftarrow_T)^*$,*
4. *T is locally confluent,*
5. *T is left-linear and non-erasing,*

then R is terminating.

As in Theorem 75 for finite TRSs conditions 2 to 5 can be verified by a finite analysis.

Proof: Condition 1 and 4 correspond to conditions 1 and 4 of Theorem 71, respectively. Since $(\to_S / (\to_T \cup \leftarrow_T))^+$ is closed under contexts and substitutions, from condition 2 we conclude $\to_R \subseteq (\to_S / (\to_T \cup \leftarrow_T))^+$, corresponding to condition

2 of Theorem 71. Finally condition 3 of Theorem 71 holds by conditions 3 and 5 and Theorem 74, choose $A = S$ and $C = \to_T^* \cdot \to_S \cdot (\to_S \cup \to_T \cup \leftarrow_T)^*$. Hence by Theorem 71 we may conclude termination of $\to_R$. □

Theorem 76 is the core of [10]. The way it is applied there is to start with a reduction order $\prec$ for which $l \succ r$ for many rules $l \to r$ of R, and some suitable T compatible with $\prec$. The TRS S is obtained from R by replacing the remaining incompatible rules $l \to r$ of R by $T(l) \to T(r)$, where $T(-)$ denotes reducing to normal form with respect to T. These new rules have to be compatible with $\prec$. Similar as described above then completion is performed until conditions 3 and 4 hold. An attempt to automate this has been made in [60].

Example 27 Let R consist of the rule $f(x,x) \to f(c,g(x))$. Since R is not left-linear, Theorem 75 can not be applied. By choosing T to consist of the rule $f(c,g(x)) \to x$ and choosing S to consist of the rule $f(x,x) \to x$, we obtain $CP(T,S) = \emptyset$, and all conditions of Theorem 76 are easily verified. Hence R is terminating.

5.4 Semantic labelling

The functional program computing the factorial can be described as a TRS as follows:

$$\begin{array}{lcl} \mathit{fact}(s(x)) & \to & \mathit{fact}(p(s(x))) * s(x) \\ p(s(0)) & \to & 0 \\ p(s(s(x))) & \to & s(p(s(x))). \end{array}$$

Termination of this program is not difficult to see: for each recursive call of *fact* the value of the argument strictly decreases. However, if we forget about the semantics of the terms representing numbers, then proving termination of the TRS is not that easy any more. The left hand side of the first rule can be embedded in the corresponding right hand side, hence the system is not simply terminating and standard techniques like recursive path order fail.

In this section we describe how termination of this kind of systems is easily proved by the technique of *semantic labelling*: given a TRS having some semantics, we introduce a labelling of the operation symbols in the TRS depending on the semantics of their arguments. We do this in such a way that termination of the original TRS is equivalent to termination of the labelled TRS. The labelled TRS has more operation symbols than the original TRS, and often more rules, sometimes even infinitely many. The original TRS can be obtained from the labelled TRS by removing all labels and removing multiple copies of rules. Although the labelled TRS is greater in some sense than the original one, in many cases termination of the labelled version is easier to prove than termination of the original one, for instance by recursive path order. In the factorial system we can label every symbol '*fact*' by the value of its argument. We obtain infinitely many distinct operation symbols 'fact_i' instead of one symbol '*fact*'; the other operation symbols do not change. The labelled TRS is obtained from the original one by replacing the first rule by infinitely many rules

$$\mathit{fact}_{i+1}(s(x)) \to \mathit{fact}_i(p(s(x))) * s(x),$$

one for every natural number i. It is easy to prove termination of this infinite labelled system by recursive path order, hence proving termination of the original factorial system.

Globally we distinguish two ways of using this technique. In the first way we choose a (quasi-)model which reflects the original semantics of the TRS, as we did

for the factorial example. In the second way we choose an artificial (quasi-)model reflecting syntactic properties that are recognized in the rewrite rules, making the technique purely syntactical. In this way we obtain termination proofs of systems like $f(f(x)) \to f(g(f(x)))$ and $f(0,1,x) \to f(x,x,x)$. This approach easily extends to proving termination modulo equations.

Recent applications of semantic labelling outside the scope of pure term rewriting are in process algebra ([27]) and in explicit substitution in λ-calculus as described by the system SUBST. The latter is presented here as Example 33. In [12] the idea of labelling has been applied to prove preservation of strong normalization in calculi for explicit substitution. In [51] a framework called *self-labelling* based on semantic labelling has been developed and applied to modularity, dummy elimination and currying. The key idea is that terms are labelled by itself.

The theory as presented here is given in more detail in [65]. In particular there the treatments for models and quasi-models are given separately while here we directly present the most general version. First we need some definitions.

Definition 77 *A Σ-algebra (A, Σ_A) is a* model *for a TRS (Σ, R) if $[\![l]\!]_\alpha = [\![r]\!]_\alpha$ for all rules $l \to r$ in R and all $\alpha : \mathcal{X} \to A$. Here $[\![-]\!]_-$ is defined as in Subsection 2.1.*

A Σ-algebra (A, Σ_A) equipped with a partial order $\leq$ is a quasi-model *for a TRS (Σ, R) if $[\![l]\!]_\alpha \geq [\![r]\!]_\alpha$ for all rules $l \to r$ in R and all $\alpha : \mathcal{X} \to A$, and f_A is weakly monotone in all arguments for all $f \in \Sigma$.*

Fix a quasi-model $(A, \Sigma_A, \leq)$. Fix for every $f \in \Sigma$ a corresponding non-empty set S_f of labels, equipped with a well-founded partial order $\leq$. This gives rise to the new signature

$$\overline{\Sigma} = \{f_s | f \in \Sigma, s \in S_f\},$$

where the arity of f_s is equal to the arity of f. An operation symbol f is called *labelled* if S_f contains more than one element. For unlabelled f the set S_f containing only one element can be left implicit, writing f instead of f_s.

Fix for every $f \in \Sigma$ a map $\pi_f : A^n \to S_f$ that is weakly monotone in all arguments, where n is the arity of f. For unlabelled f this function π_f can be left implicit. This labelling of operation symbols extends to a labelling of terms by defining $\mathsf{lab} : \mathit{Ter}(\Sigma) \times A^{\mathcal{X}} \to \mathit{Ter}(\overline{\Sigma})$ inductively by

$$\begin{aligned} \mathsf{lab}(x, \alpha) &= x, \\ \mathsf{lab}(f(t_1, \ldots, t_n), \alpha) &= f_{\pi_f([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha)}(\mathsf{lab}(t_1, \alpha), \ldots, \mathsf{lab}(t_n, \alpha)) \end{aligned}$$

for $x \in \mathcal{X}, \alpha : \mathcal{X} \to A, f \in \Sigma, t_1, \ldots, t_n \in \mathit{Ter}(\Sigma)$.

For any TRS (Σ, R) let $(\overline{\Sigma}, \overline{R})$ consist of the rules

$$\mathsf{lab}(l, \alpha) \to \mathsf{lab}(r, \alpha)$$

for all $\alpha : \mathcal{X} \to A$ and all rules $l \to r$ of R.

Let the TRS $(\overline{\Sigma}, \mathsf{Decr})$ consist of the rules

$$f_s(x_1, \ldots, x_n) \to f_{s'}(x_1, \ldots, x_n)$$

for all $f \in \Sigma$ and all $s, s' \in S_f$ satisfying $s > s'$. Here $>$ denotes the strict part of $\geq$.

If the partial order $\leq$ on A is chosen to be the equality relation, then the definitions of quasi-model and model coincide. In that case we may choose the partial order on S_f to be the equality relation too, for every $f \in \Sigma$, by which all requirements on well-foundedness and weak monotonicity are trivially fulfilled and Decr is empty.

Lemma 78 *Let $\alpha : \mathcal{X} \to A$ and let $\sigma : \mathcal{X} \to Ter(\Sigma)$. Define $\overline{\sigma} : \mathcal{X} \to Ter(\overline{\Sigma})$ by $\overline{\sigma}(x) = \mathsf{lab}(\sigma(x), \alpha)$ and $\beta : \mathcal{X} \to A$ by $\beta(x) = [\![\sigma(x)]\!]_\alpha$. Then*

$$\mathsf{lab}(t^\sigma, \alpha) \;=\; \mathsf{lab}(t, \beta)^{\overline{\sigma}}.$$

Proof: By induction on the structure of t. If t is a variable the lemma follows from the definition of $\overline{\sigma}$. If $t = f(t_1, \ldots, t_n)$ we obtain

$$\mathsf{lab}(t^\sigma, \alpha) = \mathsf{lab}(f(t_1^\sigma, \ldots, t_n^\sigma), \alpha) = f_{\pi_f([\![t_1^\sigma]\!]_\alpha, \ldots, [\![t_n^\sigma]\!]_\alpha)}(\mathsf{lab}(t_1^\sigma, \alpha), \ldots, \mathsf{lab}(t_n^\sigma, \alpha))$$

and

$$\mathsf{lab}(t, \beta)^{\overline{\sigma}} = \mathsf{lab}(f(t_1, \ldots, t_n), \beta)^{\overline{\sigma}} = f_{\pi_f([\![t_1]\!]_\beta, \ldots, [\![t_n]\!]_\beta)}(\mathsf{lab}(t_1, \beta)^{\overline{\sigma}}, \ldots, \mathsf{lab}(t_n, \beta)^{\overline{\sigma}}).$$

The labels of f are equal due to lemma 6 and the arguments are equal due to the induction hypothesis. Hence both terms are equal. □

Lemma 79 *Let $(A, \Sigma_A, \leq)$ a quasi-model for (Σ, R). Let $t, t' \in Ter(\Sigma)$ satisfy $t \to_R t'$. Then $[\![t]\!]_\alpha \geq [\![t']\!]_\alpha$ for all $\sigma : \mathcal{X} \to A$.*

Proof: If $t = l^\tau$ and $t' = r^\tau$ for some rule $l \to r$ of R and some $\tau : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ the assertion follows from lemma 6 and the definition of quasi-model.

Let $t \to_R t'$ and $[\![t]\!]_\alpha \geq [\![t']\!]_\alpha$; we still have to prove that

$$[\![f(\ldots, t, \ldots)]\!]_\alpha \geq [\![f(\ldots, t', \ldots)]\!]_\alpha$$

for all $f \in \Sigma$ and all $\alpha : \mathcal{X} \to A$. This follows from the definition of $[\![-]\!]_\_$ and the fact that f_A is weakly monotone in all coordinates. □

Lemma 80 *Let $(A, \Sigma_A, \leq)$ a quasi-model for (Σ, R). Let $t, t' \in Ter(\Sigma)$ satisfy $t \to_R t'$. Then for all $\alpha : \mathcal{X} \to A$ there is a term u over $\overline{\Sigma}$ such that*

$$\mathsf{lab}(t, \alpha) \;\to^*_{\mathsf{Decr}}\; u \;\to_{\overline{R}}\; \mathsf{lab}(t', \alpha).$$

Proof: If $t = l^\tau$ and $t' = r^\tau$ for some rule $l \to r$ of R and some $\tau : \mathcal{X} \to Ter(\Sigma)$ we obtain from lemma 78

$$\mathsf{lab}(t, \alpha) = \mathsf{lab}(l, \beta)^{\overline{\tau}} \;\to_{\overline{R}}\; \mathsf{lab}(r, \beta)^{\overline{\tau}} = \mathsf{lab}(t', \alpha),$$

for β satisfying $\beta(x) = [\![\tau(x)]\!]_\alpha$, hence the assertion holds.

Write $\Longrightarrow$ for the composition of $\to^*_{\mathsf{Decr}}$ and $\to_{\overline{R}}$. Let $t \to_R t'$ and $\mathsf{lab}(t, \alpha) \Longrightarrow \mathsf{lab}(t', \alpha)$. We still have to prove that

$$\mathsf{lab}(f(\ldots, t, \ldots), \alpha) \;\Longrightarrow\; \mathsf{lab}(f(\ldots, t', \ldots), \alpha).$$

According to lemma 79 and the fact that π_f is weakly monotone in all coordinates, we obtain $\pi_f(\ldots, [\![t]\!]_\alpha, \ldots) \;\geq\; \pi_f(\ldots, [\![t']\!]_\alpha, \ldots)$. Hence

$$\begin{array}{lll}
\mathsf{lab}(f(\ldots, t, \ldots), \alpha) & = & f_{\pi_f(\ldots, [\![t]\!]_\alpha, \ldots)}(\ldots, \mathsf{lab}(t, \alpha), \ldots) \\
& \to^*_{\mathsf{Decr}} & f_{\pi_f(\ldots, [\![t']\!]_\alpha, \ldots)}(\ldots, \mathsf{lab}(t, \alpha), \ldots) \\
& \Longrightarrow & f_{\pi_f(\ldots, [\![t']\!]_\alpha, \ldots)}(\ldots, \mathsf{lab}(t', \alpha), \ldots) \\
& = & \mathsf{lab}(f(\ldots, t', \ldots), \alpha).
\end{array}$$

□

Now we arrive at the main theorem of this section.

Theorem 81 *Let $(A, \Sigma_A, \leq)$ a quasi-model for a TRS (Σ, R). Let $\overline{R}$ and* Decr *be as above for any choice of S_f and π_f. Then (Σ, R) is terminating if and only if $(\overline{\Sigma}, \overline{R} \cup \mathsf{Decr})$ is terminating.*

Proof: Assume $\overline{R} \cup \mathsf{Decr}$ allows an infinite reduction. Since the order on S_f is well-founded for all $f \in \Sigma$, the system Decr is terminating. So the infinite reduction of $\overline{R} \cup \mathsf{Decr}$ contains infinitely many $\overline{R}$-steps. Then removing all labels yields an infinite reduction of R.

On the other hand assume that R allows an infinite reduction. Then applying lab for a fixed substitution on this infinite reduction yields an infinite reduction of $\overline{R} \cup \mathsf{Decr}$ according to lemma 80. □

An important special case of this theorem is where (A, Σ_A) a model for a TRS (Σ, R). In that case we may choose the discrete order (i.e., $x \geq y$ if and only if $x = y$) on both A and S_f. In this special case the requirements of weak monotonicity are trivially fulfilled, the notions of model and quasi-model coincide, and the TRS Decr is empty. So from Theorem 81 we conclude that (Σ, R) is terminating if and only if $(\overline{\Sigma}, \overline{R})$ is terminating.

Theorem 81 easily extends to rewriting modulo equations. For the model case this has been elaborated in detail in [65].

This section is concluded by a number of examples and exercises. We start with three examples in which the (finite) model is based on syntactical observations. A typical syntactical observation is that in a rule

$$\cdots f(g(\cdots)) \cdots \;\rightarrow\; \cdots f(h(\cdots)) \cdots$$

the f's can be forced to obtain distinct labels by choosing the images of g and h in the model to be distinct.

Example 28 The simplest example R of a terminating TRS that is not simply terminating is

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

Intuitively termination of this system is not difficult: at every step the number of operation symbols f of which the argument is again a term with head symbol f decreases. For formalizing this idea we may want to label an f-symbol by 2 if the head symbol of its argument is f, and label it by 1 otherwise. This is obtained by defining the model (A, Σ_A) by $A = \{1, 2\}$, and $f_A(x) = 2$ and $g_A(x) = 1$ for $x = 1, 2$; it is indeed a model since the interpretations of both the left hand side and the right hand side are always equal to 2. To obtain the desired labels choose $S_f = \{1, 2\}$ and π_f is the identity; choose g to be unlabelled. Then $\overline{R}$ is

$$\begin{array}{lcl} f_2(f_1(x)) & \rightarrow & f_1(g(f_1(x))) \\ f_2(f_2(x)) & \rightarrow & f_1(g(f_2(x))); \end{array}$$

the first rule is obtained if $\alpha(x) = 1$, the second if $\alpha(x) = 2$. Since we have a model we may choose discrete orders and obtain that Decr is empty. Termination of $\overline{R}$ is easily proved by counting the number of f_2 symbols, or by recursive path order. Using theorem 81 we conclude that the original system R is terminating too.

Example 29 Consider the TRS

$$f(0, 1, x) \;\rightarrow\; f(x, x, x)$$

from [62]. This system is not simply terminating as we saw in Example 11. For proving termination we want to use the observation that in the left hand side the

first and the second argument of f are distinct while in the right hand side they are equal. This distinction is made by choosing $S_f = \{a, b\}$ and $\pi_f(x, y, z) = b$ if $x = y$ and $\pi_f(x, y, z) = a$ if $x \neq y$. We still need any model in which 0 and 1 are indeed distinct; a simple one is $A = \{0, 1\}$ with $0_A = 0$, $1_A = 1$, and $f_A(x, y, z) = 0$ for $x, y, z = 0, 1$. Now we obtain the labelled system $f_a(0, 1, x) \to f_b(x, x, x)$ which is easily proved to be terminating by any standard technique.

If the TRS is extended by the rules

$$f(x, y, z) \to 2, \quad 0 \to 2, \quad 1 \to 2$$

as in the example from [21] showing non-modularity of completeness, then no non-trivial model exists any more. However, then we can extend A to a quasi-model by adjoining 2 to A and defining $0 > 2$ and $1 > 2$, and $0_A = 0$, $1_A = 1$, $2_A = 2$, and $f_A(x, y, z) = 0$ for $x, y, z = 0, 1, 2$. Choose $S_f = \{a, b\}$ and $\pi_f(x, y, z) = a$ if $x = 0$ and $y = 1$, and $\pi_f(x, y, z) = b$ otherwise, then by choosing $a > b$ the function π_f is weakly monotone in all arguments. Now $\overline{R}$ consists of the rules

$$\begin{array}{lcl} f_a(0, 1, x) & \to & f_b(x, x, x) \\ f_a(x, y, z) & \to & 2 \\ f_b(x, y, z) & \to & 2 \\ 0 & \to & 2 \\ 1 & \to & 2 \end{array}$$

and Decr consists of the rule $f_a(x, y, z) \to f_b(x, y, z)$. Termination of the extended TRS now follows from termination of $\overline{R} \cup \mathsf{Decr}$ as is easily proved by recursive path order.

Exercise 13 Prove termination of $f(0, 1, x, y) \to f(x, y, x, x)$.

Example 30 In the system

$$\begin{array}{lcl} (x * y) * z & \to & x * (y * z) \\ (x + y) * z & \to & (x * z) + (y * z) \\ x * (y + f(z)) & \to & g(x, z) * (y + a) \end{array}$$

from [18] we can force that the symbols '$*$' in the last rule get distinct labels by choosing the model $\{1, 2\}$ and defining $a_A = 1, f_A(x) = 2, \pi_*(x, y) = x +_A y = y, x *_A y = 1$ for all $x, y = 1, 2$. The labelled system is

$$\begin{array}{lcl} (x *_1 y) *_1 z & \to & x *_1 (y *_1 z) \\ (x *_1 y) *_2 z & \to & x *_1 (y *_2 z) \\ (x *_2 y) *_1 z & \to & x *_1 (y *_1 z) \\ (x *_2 y) *_2 z & \to & x *_1 (y *_2 z) \\ (x + y) *_1 z & \to & (x *_1 z) + (y *_1 z) \\ (x + y) *_2 z & \to & (x *_2 z) + (y *_2 z) \\ x *_2 (y + f(z)) & \to & g(x, z) *_1 (y + a) \end{array}$$

and is proved terminating using recursive path order: give $*_1$ a lexicographic status, choose $*_2$ to be greater than all the other symbols and choose $*_1 > +$.

In the next example the model corresponds to the natural semantics of the rewrite system.

Example 31 In the factorial system as presented in the beginning of this section choose $A = \mathbb{N}$, $0_A = 0$, $s_A(x) = x + 1$, $p_A(0) = 0$, and $p_A(x) = x - 1$ for $x > 0$. Further choose $x *_A y = x * y$ and $fact_A(x) = x!$. Clearly (A, Σ_A) is a model for the

system; by labelling $fact$ with the naturals and choosing $\pi_{fact}(x) = x$ we get the labelled version

$$\begin{array}{lcl} fact_{i+1}(s(x)) & \to & fact_i(p(s(x))) * s(x) \\ p(s(0)) & \to & 0 \\ p(s(s(x))) & \to & s(p(s(x))) \end{array}$$

in which the first line stands for infinitely many rules, one for every $i \in \mathbb{N}$. By $fact_{i+1} > fact_i > * > p > s$ this labelled system is proved to be terminating by recursive path order.

Exercise 14 Prove termination of

$$\begin{array}{lcl} f(0) & \to & 0 \\ f(s(x)) & \to & s(s(f(p(s(x))))) \\ p(s(0)) & \to & 0 \\ p(s(s(x))) & \to & s(p(s(x))). \end{array}$$

In the last three examples we use quasi-models instead of models.

Example 32 We prove that for every $k, n \geq 0$ the TRS

$$f(g(f(x))) \to f^k(g^n(x))$$

is simply terminating. In Proposition 35 we already used this fact for $k = 3, n = 2$. By Proposition 21 simple termination is equivalent to termination of the system extended by the embedding rules $f(x) \to x, g(x) \to x$. By adding some non-empty context to any reduction in this system, we see that every application of $g(x) \to x$ is either an application of $f(g(x)) \to f(x)$ or an application of $g(g(x)) \to g(x)$. Hence simple termination of the original rule is equivalent to termination of the system

$$\begin{array}{rcl} f(g(f(x))) & \to & f^k(g^n(x)) \\ f(x) & \to & x \\ f(g(x)) & \to & f(x) \\ g(g(x)) & \to & g(x). \end{array}$$

For $n = 0$ the statement is simple; we assume $n > 0$. By choosing the quasi-model $A = \{0, 1\}$ with $1 > 0$, $f_A(n) = 1, g_A(n) = 0$ for $n = 0, 1$, and labelling g with $\pi_g(n) = n$ for $n = 0, 1$, we have to prove termination of the labelled system

$$\begin{array}{rcl} f(g_1(f(x))) & \to & f^k(g_0^n(x)) \\ f(g_1(f(x))) & \to & f^k(g_0^{n-1}(g_1(x))) \\ f(x) & \to & x \\ f(g_0(x)) & \to & f(x) \\ f(g_1(x)) & \to & f(x) \\ g_0(g_0(x)) & \to & g_0(x) \\ g_0(g_1(x)) & \to & g_1(x) \\ g_1(x) & \to & g_0(x) \end{array}$$

Termination of this system follows from compatibility with the recursive path order for a precedence $\prec$ satisfying $g_1 \succ g_0$ and $g_1 \succ f$.

Example 33 Let $\circ$ and $\cdot$ be binary symbols, λ a unary symbol, and 1, id and $\uparrow$

constants. Consider the TRS

$$\begin{array}{lcl}
\lambda(x) \circ y & \to & \lambda(x \circ (1 \cdot (y \circ \uparrow))) \\
(x \cdot y) \circ z & \to & (x \circ z) \cdot (y \circ z) \\
(x \circ y) \circ z & \to & x \circ (y \circ z) \\
id \circ x & \to & x \\
1 \circ id & \to & 1 \\
\uparrow \circ id & \to & \uparrow \\
1 \circ (x \cdot y) & \to & x \\
\uparrow \circ (x \cdot y) & \to & y,
\end{array}$$

named σ_0 in [14], which is essentially the same as the system SUBST in [32]. This system describes the process of substitution in combinatory categorical logic. Here 'λ' corresponds to currying, '$\circ$' to composition, 'id' to the identity, '$\cdot$' to pairing and '1' and '$\uparrow$' to projections. The original termination proof of SUBST in [32] is very complicated; the same holds for the proof by [14]. For both papers the termination proof of this particular system is the main result. The result implies termination of the process of explicit substitution in untyped λ-calculus; an overview of this approach to explicit substitution is given in [1]. In [64] the technique of distribution elimination (see Theorem 68) was developed to prove simple termination of σ_0. Define the TRS R to consist of the first three rules of σ_0 and the embedding rules

$$\lambda(x) \to x, \;\; x \circ y \to x, \;\; x \circ y \to y, \;\; x \cdot y \to x, \;\; x \cdot y \to y.$$

Clearly simple termination of σ_0 is equivalent to termination of R. Here we prove termination of R by means of theorem 81. As the quasi-model we choose the natural numbers (including 0) and

$$\lambda_A(x) = x + 1, \;\; x \circ_A y = x + y, \;\; x \cdot_A y = max(x, y), \;\; 1_A = \uparrow_A = \; 0.$$

One easily checks that this is indeed a quasi-model for R. Only the symbol $\circ$ is labelled; it is labelled by its own value. More precisely, we choose $S_\circ$ to be the natural numbers and $\pi_\circ(x, y) = x + y$. Now the system $\overline{R} \cup \mathsf{Decr}$ reads

$$\begin{array}{lcll}
\lambda(x) \circ_i y & \to & \lambda(x \circ_j (1 \cdot (y \circ_k \uparrow))) & \text{for values } i > j \text{ and } i > k \\
(x \cdot y) \circ_i z & \to & (x \circ_j z) \cdot (y \circ_k z) & \text{for values } i \geq j \text{ and } i \geq k \\
(x \circ_j y) \circ_i z & \to & x \circ_i (y \circ_k z) & \text{for values } i \geq j \text{ and } i \geq k \\
\lambda(x) & \to & x & \\
x \circ_i y & \to & x & \text{for all values } i \\
x \circ_i y & \to & y & \text{for all values } i \\
x \cdot y & \to & x & \\
x \cdot y & \to & y & \\
x \circ_i y & \to & x \circ_j y & \text{for all values } i > j.
\end{array}$$

By choosing the well-founded precedence

$$\circ_i > \circ_j \;\text{ for } i > j, \quad \circ_i > \lambda, \; \circ_i > \cdot \,, \; \circ_i > 1, \; \circ_i > \uparrow \;\text{ for all } i$$

termination is easily proved by the lexicographic path order. Now theorem 81 yields termination of R, and hence simple termination of σ_0.

Example 34 Let $|$ and $\cdot$ be binary symbols and 0 a constant. Consider the TRS

$$\begin{array}{lcl}
x \cdot 0 & \to & x \\
0 \cdot x & \to & x \\
x \mid 0 & \to & x \\
0 \mid x & \to & 0 \\
x \mid x & \to & 0 \\
(x \cdot y) \mid z & \to & (x \mid z) \cdot (y \mid (z \mid x)) \\
z \mid (x \cdot y) & \to & (z \mid x) \mid y.
\end{array}$$

These rules are the directed versions of some well-known equations that hold for reductions in lambda calculus or orthogonal rewrite systems; see e.g. Prop. 12.2.2 in Barendregt [84]. We prove termination of R by means of theorem 81. As the quasi-model we choose the strictly positive integers, and

$$0_A = 1, \quad x \cdot_A y = x + y, \quad x \mid_A y = x.$$

One easily checks that this is indeed a quasi-model for R. Only the symbol $|$ is labelled; it is labelled by the sum of the values of its arguments: $S_|$ consists of the strictly positive integers and $\pi_|(x, y) = x + y$. Now the system $\overline{R} \cup \mathsf{Decr}$ reads

$$\begin{array}{lcll}
x \mid_i y & \to & x \mid_j y & \text{for all values } i > j \\
x \cdot 0 & \to & x & \\
0 \cdot x & \to & x & \\
x \mid_i 0 & \to & x & \text{for all values } i \\
0 \mid_1 x & \to & 0 & \\
x \mid_i x & \to & 0 & \text{for all values } i \\
(x \cdot y) \mid_i z & \to & (x \mid_j z) \cdot (y \mid_k (z \mid_j x)) & \\
z \mid_i (x \cdot y) & \to & (z \mid_j x) \mid_k y, &
\end{array}$$

where in the last two lines i, j, k run over all possible values of

$$i = [\![x]\!]_\alpha + [\![y]\!]_\alpha + [\![z]\!]_\alpha,$$

$$j = [\![x]\!]_\alpha + [\![z]\!]_\alpha,$$

$$k = [\![y]\!]_\alpha + [\![z]\!]_\alpha,$$

always satisfying $i > j$ and $i > k$. By choosing the well-founded precedence

$$|_i > |_j \text{ for all } i > j, \quad |_i > \cdot > 0 \text{ for all } i$$

termination is immediate by any version of recursive path order.

5.5 The dependency pair method

In this section we discuss the dependency pair method developed by Arts and Giesl ([4, 3]). By this method termination of a wide variety of TRSs can be proved automatically, including many TRSs that are not simply terminating. Roughly speaking, a dependency pair is a pair of terms extracted from the rewrite rules that describe how arguments of defined symbols can be rewritten, where defined symbols are the symbols that occur as root symbols of left hand sides. One of their key results states that termination of a TRS can be concluded if a particular quasi-order $\geq$ exists such that $l \geq r$ for all rewrite rules $l \to r$ and $s \geq t$ for all dependency pairs $\langle s, t \rangle$, while $s > t$ is required for only some essential dependency pairs $\langle s, t \rangle$. The basic way of finding such a quasi-order $\geq$ is defining $s \geq t \;\Leftrightarrow\; N(s) \succ_{rpo} N(t)$, where N denotes the normal form with respect to a suitable recursive scheme. Typically, by N some of the arguments of some function symbols are ignored.

Here we present an outline of the approach; for more details and extensions of the method and for proofs of the theorems we refer to [4, 3].

Definition 82 • *For a TRS (Σ, R) a symbol $f \in \Sigma$ is called a defined symbol if f is the root symbol of a left hand side of a rule of R. Symbols $f \in \Sigma$ are written in lower case letters, for every defined symbol $f \in \Sigma$ a new capitalized symbol F is added having the same arity as f.*

- *If* $f(s_1,\ldots,s_n) \to C[g(t_1,\ldots,t_m)]$ *is a rule in* R *and* g *is a defined symbol of* R*, then*
$$\langle F(s_1,\ldots,s_n), G(t_1,\ldots,t_m)\rangle$$
is called a dependency pair of R*.*

- *An infinite sequence* $(\langle s_i,t_i\rangle)_{i=1,2,3,\ldots}$ *of dependency pairs of a TRS* R *is called an infinite* R*-chain if subsititutions* σ_i *exist such that* $t_i^{\sigma_i} \to_R^* s_{i+1}^{\sigma_{i+1}}$ *for every* $i = 1,2,3,\ldots$.

This definition is motivated by the following theorem.

Theorem 83 *A TRS* (Σ,R) *is terminating if and only if it does not admit an infinite* R*-chain.*

Example 35 As in Example 7 consider the TRS R consisting of the rule:
$$f(f(x)) \;\to\; f(g(f(x))).$$
Here f is the only defined symbol, hence the signature is extended by one unary symbol F. In the right hand side two copies of the defined symbol f occur, hence there are two dependency pairs:
$$\langle F(f(x)), F(g(f(x)))\rangle \quad \text{and} \quad \langle F(f(x)), F(x)\rangle.$$
Assume that $(\langle s_i,t_i\rangle)_{i=1,2,3,\ldots}$ is an infinite R-chain. Since no substitutions σ,τ exist satisfying $F(g(f(x))^\sigma \to_R^* F(f(x))^\tau$, we conclude that $s_i = F(f(x))$ and $t_i = F(x)$ for all $i = 1,2,3,\ldots$. Now from $F(x)^{\sigma_i} \to_R^* F(f(x))^{\sigma_{i+1}}$ we conclude that x^{σ_i} contains one more f-symbol then $x^{\sigma_{i+1}}$ for every $i = 1,2,3,\ldots$, contradiction. Hence by Theorem 83 we have proved that R is terminating.

Finding the dependency pairs of a finite TRS is easily done automatically. The search for infinite chains is much harder; since establishing termination is undecidable this search for infinite chains is undecidable too by Theorem 83. The goal now is to develope techniques for automatically proving the non-existence of infinite chains for a wide variety of TRSs, and hence termination. A first step is to remove dependency pairs that are not essential for the existence of infinite chains. Dependency pairs can be seen as the nodes of a dependency graph, where an arc from a dependency pair $\langle s,t\rangle$ to dendency pair $\langle u,v\rangle$ is drawn if substitutions σ,τ exist satisfying $t^\sigma \to_R^* u^\tau$. In Example 35 the dependency graph consists of two nodes $\langle F(f(x)), F(g(f(x)))\rangle$ and $\langle F(f(x)), F(x)\rangle$, and two arcs: one from $\langle F(f(x)), F(x)\rangle$ to $\langle F(f(x)), F(g(f(x)))\rangle$ and one from $\langle F(f(x)), F(x)\rangle$ to itself.

By definition any infinite chain gives rise to an infinite path in the dependency graph. Since the dependency graph is finite, one can prove that an infinite chain exists if and only an infinite chain exists only involving dependency pairs that are on a cycle of the dependency graph. Hence for using Theorem 83 for proving termination all dependency pairs that are not on a cycle of the dependency graph may be ignored. In Example 35 this means that the dependency pair $\langle F(f(x)), F(g(f(x)))\rangle$ may be ignored.

However, for establishing the full dependency graph there is a serious problem: for arbitrary dependency pairs $\langle s,t\rangle$ and $\langle u,v\rangle$ it is undecidable whether substitutions σ,τ exist satisfying $t^\sigma \to_R^* u^\tau$. Hence in general we can not compute the exact dependency graph. The best we can do is to approximate the dependency graph by an approximated dependency graph which is a supergraph of the dependency graph, and for which the approximated dependency graph can be computed automatically. This is done in the following way.

Definition 84 *The function* cap *is defined inductively by*

$$\begin{array}{rcll} \mathsf{cap}(x) & = & x & \\ \mathsf{cap}(f(t_1,\ldots,t_n)) & = & y & \text{if } f \text{ is a defined symbol} \\ \mathsf{cap}(f(t_1,\ldots,t_n)) & = & f(\mathsf{cap}(t_1),\ldots,\mathsf{cap}(t_n)) & \text{if } f \text{ is not a defined symbol;} \end{array}$$

here y is some fixed variable. The function lin *renames all variables of a term in such a way that all variables, including multiple occurrences of the same variable, are replaced by a fresh variable.*

The approximated dependency graph of a TRS R is the graph of which the nodes are the dependency pairs of R and there is an arc from $\langle s,t\rangle$ to $\langle u,v\rangle$ if and only if substitutions σ,τ exist satisfying $(\mathsf{lin}(\mathsf{cap}(t))^\sigma = u^\tau$.

Note that $\mathsf{lin}(t)$ is a linear term for every term t. For example, if f and g are defined symbols and h is not a defined symbol then for $t = h(x, f(x,y), h(g(z), x, g(x)))$ we have $\mathsf{cap}(t) = h(x, y, h(y, x, y))$ and $\mathsf{lin}(\mathsf{cap}(t)) = h(x_1, x_2, h(x_3, x_4, x_5))$.

Note that establishing whether there is an arc from one dependency pairs to another one is merely unification, hence establishing the approximated dependency graph of a TRS is easily implemented.

On the other hand one can prove that if substitutions σ,τ exist satisfying $t^\sigma \to_R^* u^\tau$, then indeed ($\mathsf{lin}(\mathsf{cap}(t)$ and u unify. Hence indeed the approximated dependency graph is a supergraph of the dependency graph. As a consequence we have that for proving termination by Theorem 83 we may ignore all dependency pairs that are not on a cycle of the approximated dependency graph. Establishing which nodes of a graph are on a cycle can be done by standard algorithms, hence removing these redundant dependency pairs can be done fully automatically.

Indeed if we apply this approach to Example 35, then the dependency pair $\langle F(f(x)), F(g(f(x))\rangle$ is omitted automatically.

In order to extend these observations to a method that can be used for automatically proving termination of a TRS we still need a method for automatically proving the non-existence of infinite R-chains. This is done by proving compatibility with a suitable kind of quasi-ordering. First we need a definition.

Definition 85 *A quasi-ordering $\leq$ on terms is called a* reduction quasi-ordering *if*

- *$<$ is well-founded;*
- *if $t \geq s$ then $t^\sigma \geq s^\sigma$ for all substitutions σ;*
- *if $t > s$ then $t^\sigma > s^\sigma$ for all substitutions σ;*
- *if $t \geq s$ then $f(\ldots,t,\ldots) \geq f(\ldots,s,\ldots)$ for all $f(\ldots)$.*

Here $s < t \;\Leftrightarrow\; (s \leq t \wedge \neg(t \leq s))$, and $\geq$ and $>$ are the inverses of $\leq$ and $<$, respectively.

Theorem 86 *Let (Σ, R) be a TRS and let $\leq$ be a reduction quasi-ordering such that*

- *$l \geq r$ for all rules $l \to r$ in R, and*
- *$s \geq t$ for all dependency pairs $\langle s,t\rangle$ on a cycle in the approximated dependency graph, and*
- *every cycle in the approximated dependency graph contains at least one dependency pairs $\langle s,t\rangle$ satisfying $s > t$.*

Then (Σ, R) is terminating.

Proof: (sketch)

Assume (Σ, R) is not terminating. Then by Theorem 83 it admits an infinite R-chain $(\langle s_i, t_i\rangle)_{i=1,2,3,\ldots}$. Since $(\langle s_i, t_i\rangle, \langle s_{i+1}, t_{i+1}\rangle)$ is an arc of the approximated dependency graph for every i, we conclude that $\langle s_i, t_i\rangle$ is on a cycle in the approximated dependency graph for every $i \geq N$ for some N, while $s_i > t_i$ occurs infinitely often by the third condition of the theorem. By the definition of R-chain substitutions σ_i exist such that $t_i^{\sigma_i} \to_R^* s_{i+1}^{\sigma_{i+1}}$ for every $i = 1, 2, 3, \ldots$. Using the first and second condition of the theorem we obtain

$$s_N^{\sigma_N} \geq t_N^{\sigma_N} \geq s_{N+1}^{\sigma_{N+1}} \geq t_{N+1}^{\sigma_{N+1}} \geq s_{N+2}^{\sigma_{N+2}} \geq t_{N+2}^{\sigma_{N+2}} \geq \cdots$$

while $s_i^{\sigma_i} > t_i^{\sigma_i}$ occurs infinitely often, contradicting well-foundedness. □

In order to complete the method of proving termination by means of dependency pairs we still have to describe a way of finding suitable reduction quasi-orderings. This is a combination of recursive path order and recursive program schemes. Recall from Definition 61 that a recursive program scheme (RPS) is a TRS in which all left hand sides of the rules have distinct root symbols, and all of these left hand sides are of the shape $f(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are distinct variables. Typically here we will use RPSs in which the right hand sides are single variables or terms of the shape $g(x_{i_1}, \ldots, x_{i_k})$. Every RPS is confluent since it is orthogonal. For a terminating RPS S and a term t we write $S(t)$ for the unique normal form of t with respect to S. For any terminating RPS S and any recursive path order $\prec_{rpo}$ define

$$s \leq t \quad \Leftrightarrow \quad (S(s) = S(t) \vee S(s) \prec_{rpo} S(t)).$$

It is not difficult to prove that $\leq$ defined in this way is a reduction quasi-ordering. The dependency pair method of proving termination of a given TRS now consists of searching for a terminating RPS S and a recursive path order $\prec_{rpo}$ such that the corresponding reduction quasi-ordering $\leq$ satisfies the requirements of Theorem 86. Restricting to RPSs in which the right hand sides are single variables or linear terms of the shape $g(x_{i_1}, \ldots, x_{i_k})$, only finitely many choices are possible, yielding a full decision procedure for applicability of this method. The intuition of S is that particular arguments of particular operations, or the operations symbols themselves are systematically removed before $\prec_{rpo}$ is applied. We illustrate this for Example 35: the TRS R consisting of the rule

$$f(f(x)) \ \to \ f(g(f(x))).$$

From the two dependency pairs $\langle F(f(x)), F(g(f(x))\rangle$ and $\langle F(f(x)), F(x)\rangle$ the first one is removed since it is not on a cycle one the approximated dependency graph. Theorem 86 yield the requirements

$$f(f(x)) \ \geq \ f(g(f(x)))$$

$$F(f(x)) \ > \ F(x).$$

These are easily fulfilled by choosing S to consist of the single rule $g(x) \to x$, yielding

$$S(f(f(x))) = f(f(x)) = S(f(g(f(x)))), \quad \text{and}$$

$$S(F(f(x))) = F(f(x)) \succ_{rpo} F(x) = S(F(x))$$

for any $\prec_{rpo}$, proving termination.

Before we give a more serious example we summarize the dependency pair method:

- For a given TRS, establish all dependency pairs.

- Next, establish the approximated dependency graph and its cycles; remove all dependency pairs that are not on a cycle.
- Search for a terminating RPS S and a recursive path order $\prec_{rpo}$ such that the corresponding reduction quasi-ordering $\leq$ satisfies the requirements of Theorem 86.

If the search in the last step succeeds, then termination of the TRS has been proved. In the last step $\prec_{rpo}$ may be replaced by any order proving termination of a TRS; in this sence the dependency pair approach fits in the framework of a transformational method.

Example 36 Let R be the TRS from [4] consisting of the following four rules

$$\begin{array}{rcl} m(x,0) & \to & x \\ m(s(x),s(y)) & \to & m(x,y) \\ q(0,s(x)) & \to & 0 \\ q(s(x),s(y)) & \to & s(q(m(x,y),s(y))). \end{array}$$

Interpreting s as successor, $m(x,y)$ as $x-y$ and $q(x,y)$ as x/y this system describes division in the natural numbers. For instance one can derive

$$q(s^{21}(0),s^3(0)) \to_R^* s^7(0).$$

Termination of R is not trivial; in particular the self-embedding reduction

$$q(s(0),s(s(0))) \ \to_R \ s(q(m(0,s(0)),s(s(0))))$$

shows that R is not simply terminating. We now show that the dependency pair method successfully applies to R.

The defined symbols are m and q, and the dependency pairs are

$$\langle M(s(x),s(y)),M(x,y)\rangle,$$
$$\langle Q(s(x),s(y)),M(x,y)\rangle, \quad \text{and}$$
$$\langle Q(s(x),s(y)),Q(m(x,y),s(y))\rangle.$$

The approximated dependency graph has these three dependency pairs as its nodes. Since the function cap does not affect the symbols M and Q it is easily seen (and automatically derived) that there are four arcs in the approximated dependency graph:

- one from $\langle Q(s(x),s(y)),Q(m(x,y),s(y))\rangle$ to itself;
- one from $\langle Q(s(x),s(y)),Q(m(x,y),s(y))\rangle$ to $\langle Q(s(x),s(y)),M(x,y)\rangle$;
- one from $\langle Q(s(x),s(y)),M(x,y)\rangle$ to $\langle M(s(x),s(y)),M(x,y)\rangle$;
- one from $\langle M(s(x),s(y)),M(x,y)\rangle$ to itself.

Hence the dependency pair $\langle Q(s(x),s(y)),M(x,y)\rangle$ is not on a cycle in the approximated dependency graph, and may be removed. The remaining two dependency pairs are both on the cycle implied by the arc to itself; there are no other cycles. Hence we have to search for a terminating RPS S and a recursive path order $\prec_{rpo}$ such that the corresponding reduction quasi-ordering $\leq$ satisfies the following requirements:

$$\begin{array}{rcl} m(x,0) & \geq & x \\ m(s(x),s(y)) & \geq & m(x,y) \\ q(0,s(x)) & \geq & 0 \\ q(s(x),s(y)) & \geq & s(q(m(x,y),s(y))) \\ M(s(x),s(y)) & > & M(x,y) \\ Q(s(x),s(y)) & > & Q(m(x,y),s(y)). \end{array}$$

These requirements are fulfilled by choosing S to consist of the rule $m(x,y) \to x$ and observing

$$\begin{array}{rcl} x & = & x \\ s(x) & \succ_{rpo} & x \\ q(0,s(x)) & \succ_{rpo} & 0 \\ q(s(x),s(y)) & \succ_{rpo} & s(q(x,s(y))) \\ M(s(x),s(y)) & \succ_{rpo} & M(x,y) \\ Q(s(x),s(y)) & \succ_{rpo} & Q(x,s(y)) \end{array}$$

for any precedence $\prec$ satisfying $s \prec q$. Hence by Theorem 86 we have proved termination of R.

5.6 Type introduction

The idea of type introduction is that the one-sorted TRS for which termination has to be proven, is transformed into a many-sorted TRS having the same rules, but for which termination is easier to prove. This fits in the framework of non-termination preserving transformations; indeed our main theorem states that under some conditions termination of the original one-sorted TRS follows from termination of the many-sorted TRS.

First we recall some standard terminology. Let S be a finite set representing the set of types or sorts. An *S-sorted set* X is defined to be a family of sets $(X_s)_{s\in S}$. For S-sorted sets X and Y, an *S-sorted map* $\phi : X \to Y$ is defined to be a family of maps $(\phi_s : X_s \to Y_s)_{s\in S}$.

By S^* we denote the set of finite sequences of elements of S, including the empty sequence. Let Σ be a set of symbols, the *operation symbols*. For every operation symbol an *arity* and a *sort* is given, described by functions

$$ar : \Sigma \to S^* \quad \text{and} \quad st : \Sigma \to S.$$

Let $\mathcal{X}$ be an S-sorted set of symbols, the *variables*, where the sets $\mathcal{X}_s$ are pairwise disjoint. We define the S-sorted set $Ter(\Sigma)$ of *S-sorted terms* inductively by

- $\mathcal{X}_s \subseteq Ter(\Sigma)_s$ for $s \in S$;
- $f(t_1,\ldots,t_n) \in Ter(\Sigma)_s$ for $f \in \Sigma$ with $ar(f) = (s_1,\ldots,s_n)$ and $st(f) = s$, and $t_i \in Ter(\Sigma)_{s_i}$ for $i = 1,\ldots,n$.

An S-sorted *term rewriting system* (TRS) is defined to be an S-sorted set of rules: a rule $l \to r$ of sort s consists of two terms l and r of sort s for which l is no variable and r contains no variables that do not occur in l. The corresponding reduction relation is defined as expected. By *many-sorted* we mean S-sorted for $\#S > 1$, the case $\#S = 1$ is called *one-sorted* and corresponds to the usual notion of terms and term rewriting.

By removing all sort information every many-sorted term can be mapped to a one-sorted term as follows. Let Σ' be the set of symbols obtained by adding a prime ($'$) to every symbol of Σ. For $f \in \Sigma$ with $ar(f) = (s_1,\ldots,s_n)$ we define the arity of $f' \in \Sigma'$ to be n. In this way Σ' defines a one-sorted signature. Since there is only one sort there is no need for an explicit notation for the sort. We choose $\mathcal{X}' = \bigcup_{s\in S}\mathcal{X}_s$ to be the set of one-sorted variable symbols; recall that the sets $\mathcal{X}_s$ are assumed to be pairwise disjoint.

Every term over Σ of any sort can be mapped to a term over Σ' by adding prime symbols to all operation symbols. This type elimination map

$$\Theta : \bigcup_{s\in S} Ter(\Sigma)_s \quad \to \quad Ter(\Sigma')$$

is inductively defined by

- $\Theta(x) = x$ for every $x \in \mathcal{X}_s$, for every $s \in S$;
- $\Theta(f(t_1, \ldots, t_n)) = f'(\Theta(t_1), \ldots, \Theta(t_n))$ for all $f \in \Sigma$ and terms $t_1, \ldots, t_n$ of the right sort.

Ignoring prime symbols, the set $\bigcup_{s \in S} Ter(\Sigma)_s$ can be considered as a subset of $Ter(\Sigma')$, namely the set of well-typed terms.

The type elimination map Θ is defined on TRSs in an obvious way: for any many-sorted TRS R the one-sorted TRS $\Theta(R)$ is defined to consist of the rules $\Theta(l) \to \Theta(r)$ for the rules $l \to r$ from R. One easily observes that for $t_1, t_2 \in Ter(\Sigma)_s$:

$$t_1 \to_R t_2 \quad \Leftrightarrow \quad \Theta(t_1) \to_{\Theta(R)} \Theta(t_2).$$

A property of binary relations is called *persistent* if for every many-sorted TRS R the property holds for R if and only if it holds for $\Theta(R)$.

The notion of persistence is closely related to the notion of *modularity* as it has been studied extensively in e.g. [49, 63, 58]. A property of one-sorted TRSs is called *modular* if for every pair of (one-sorted) TRSs R_1 and R_2 with disjoint sets of operation symbols the property holds for both R_1 and R_2 if and only if it holds for $R_1 \oplus R_2$. Here $R_1 \oplus R_2$ denotes the union of both TRSs; it is a one-sorted TRS over the disjoint union of the sets of operation symbols. By choosing one sort for all operations occurring in R_1 and another sort for all operations occurring in R_2, it is not difficult to see that for properties like confluence, weak confluence, termination and weak normalization persistence implies modularity; for details we refer to [64]. In particular, the basic example from [62] showing that termination is not modular straightforwardly leads to an example that termination is not a persistent property as follows. Let $S = \{s_1, s_2\}$; the following variables and operation symbols are defined:

- x is a variable of sort s_1;
- y, z are variables of sort s_2;
- $0, 1$ are constants of sort s_1;
- f is an operation symbol of sort s_1 and arity (s_1, s_1, s_1);
- g is an operation symbol of sort s_2 and arity (s_2, s_2).

Let the S-sorted TRS R consist of the following rules:

$$\begin{array}{lcl} f(0,1,x) & \to & f(x,x,x) \\ g(y,z) & \to & y \\ g(y,z) & \to & z. \end{array}$$

The S-sorted TRS R is terminating; for sort s_1 this was proved in Example 9 and for sort s_2 this is trivial. On the other hand

$$f(g(0,1), g(0,1), g(0,1)) \to f(0, g(0,1), g(0,1)) \to f(0,1, g(0,1)) \to$$

$$f(g(0,1), g(0,1), g(0,1)) \to \cdots$$

is an infinite reduction in $\Theta(R)$. This implies that termination is not a persistent property.

However, we can define a particular class of many-sorted TRSs such that termination is persistent for that class. Recall that a reduction rule is called *collapsing* if its right hand side is a single variable, and *duplicating* if some variable occurs more often in the right hand side than in the left hand side. In the above example the first rule is duplicating and the second and the third rules are collapsing.

It has been shown by [58] that termination is modular for the class of one-sorted TRSs without collapsing rules and also in the class of one-sorted TRSs without duplicating rules. The following theorem generalizes this result:

Theorem 87 *Termination is persistent for the class of many-sorted TRSs not containing both collapsing and duplicating rules.*

Any infinite reduction of R is trivially translated to an infinite reduction of $\Theta(R)$. As a consequence, termination of $\Theta(R)$ implies termination of R. The difficult part is the converse: assume termination of R and derive termination of $\Theta(R)$. For a proof we refer to [64].

Theorem 87 can be used as a tool for proving termination of TRS as follows. Given a (one-sorted) TRS not containing both collapsing and duplicating rules, we try to find a many-sorted TRS R in such a way that $\Theta(R)$ coincides with the original one-sorted TRS. Intuitively this means that we look for a typing of the TRS in such a way that all rules are well-typed. This process is called *type introduction*, and can be done straightforwardly as follows. First try to choose distinct incoming and outcoming sorts for all symbols. Next the requirements that all left hand sides and right hand sides are well-sorted terms, and every left hand side has the same sort as the corresponding right hand side, force that many of these sorts are equal. If all sorts are forced to coincide in this way, the method does not apply. On the other hand, if some distinct sorts remain then we get a many-sorted TRS, and Theorem 87 states that for proving termination of the original one-sorted system it suffices to prove termination of the many-sorted system. The latter can be much simpler as will be illustrated now by some examples.

Example 37 As in Example 9 consider the rule

$$f(0,1,x) \quad \rightarrow \quad f(x,x,x).$$

Now termination can be proved by considering the typing in which $0, 1, x$ have sort s_1, and f has sort s_2 and arity (s_1, s_1, s_1). Then this many-sorted system is trivially terminating since at most one reduction step can be done. From Theorem 87 we conclude that also the original one-sorted system is terminating.

Example 38 Consider the TRS

$$\begin{aligned} s(x) + (y + w) &\rightarrow x + (s(s(y)) + w) \\ s(x) + (y + (z + w)) &\rightarrow x + (z + (y + w)) \end{aligned}$$

from [34]. The termination proof can be given via the typing

$$x, y, z : s_1, \quad w : s_2, \quad s : s_1 \rightarrow s_1, \quad + : (s_1, s_2) \rightarrow s_2.$$

After adding one constant 0 of sort s_1 and one constant nill of sort s_2, many-sorted terms of sort s_2 then represent lists of natural numbers. It is easy to check that after every rewrite step the length of such a list does not change, and the list itself lexicographically decreases. Hence the many-sorted system is terminating; from Theorem 87 we conclude that also the original one-sorted system is terminating.

From these examples we see that type introduction provides a far more powerful tool for proving termination than the similar modularity result.

Theorem 87 plays an important role in the proof of undecidability of simple termination for single rewrite rules in [50]. The same holds for later undecidability results for single rewrite rules in [31].

We saw that termination is not persistent for many-sorted TRSs containing both collapsing and duplicating rules. In [2] it has been shown that termination

is persistent for the class of TRSs in which all variables have the same sort. This means that for TRSs containing both collapsing and duplicating rules termination still can be concluded from termination of a many-sorted version in case all variables in this many-sorted version have the same sort.

References

[1] Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J. Explicit substitutions. *Journal of Functional Programming 1*, 4 (1991), 375–416.

[2] Aoto, T. A proof of the conjecture of Zantema on a persistent property of term rewriting systems. Tech. Rep. IS-RR-98-0008F, School of Information Science, Japan Advanced Institute of Science and Technology, 1998.

[3] Arts, T., and Giesl, J. Automatically proving termination where simplification orderings fail. In *Proceedings Theory and Practice of Software Development (TAPSOFT97, CAAP/FASE)* (1997), M. Bidoit and M. Dauchet, Eds., vol. 1214 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 261–273.

[4] Arts, T. H. J. J. *Automatically Proving Termination and Innermost Normalisation of Term Rewriting Systems.* PhD thesis, Utrecht University, 1997.

[5] Bachmair, L. Associative-commutative reduction orderings. *Information Processing Letters 43* (1992), 21–27.

[6] Bachmair, L., and Dershowitz, N. Commutation, transformation and termination. In *Proceedings of the 8th International Conference on Automated Deduction (CADE8)* (1986), J. Siekmann, Ed., vol. 230 of *Lecture Notes in Computer Science*, Springer, pp. 5–20.

[7] Bachmair, L., and Plaisted, D. Termination orderings for associative-commutative rewrite systems. *Journal of Symbolic Computation 1* (1985), 329–349.

[8] Baeten, J. C. M., and Weijland, W. P. *Process Algebra*, vol. 18 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1990.

[9] Bellegarde, F., and Lescanne, P. Transformation orderings. In *Proceedings of the 12th Colloquium on Trees in Algebra and Programming (CAAP)* (1987), vol. 249 of *Lecture Notes in Computer Science*, Springer, pp. 69–80.

[10] Bellegarde, F., and Lescanne, P. Termination by completion. *Applicable Algebra in Engineering, Communication and Computing 1*, 2 (1990), 79–96.

[11] Ben-Cherifa, A., and Lescanne, P. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computing Programming 9*, 2 (1987), 137–159.

[12] Bloo, R., and Geuvers, H. Explicit substitution: on the edge of strong normalization. *Theoretical Computer Science 204* (1998). To appear.

[13] Caron, A. C. Linear bounded automata and rewrite systems: influence of initial configurations on decision properties. In *Proceedings of the Colloquium on Trees in Algebra and Programming* (1991), vol. 493 of *Lecture Notes in Computer Science*, Springer, pp. 74–89.

[14] Curien, P.-L., Hardin, T., and Ríos, A. Strong normalization of substitutions. In *Proceedings Mathematical Foundations of Computer Science 1992* (1992), I. M. Havel and V. Koubek, Eds., vol. 629 of *Lecture Notes in Computer Science*, Springer, pp. 209–217.

[15] Dauchet, M. Simulation of Turing machines by a regular rewrite rule. *Theoretical Computer Science 103*, 2 (1992), 409–420. (Preliminary version appeared in Proceedings of RTA89, Lecture Notes in Computer Science 355, Springer, 1989.).

[16] Dershowitz, N. A note on simplification orderings. *Information Processing Letters 9*, 5 (1979), 212–215.

[17] Dershowitz, N. Orderings for term rewriting systems. *Theoretical Computer Science 17*, 3 (1982), 279–301.

[18] Dershowitz, N. Termination of rewriting. *Journal of Symbolic Computation 3*, 1 and 2 (1987), 69–116.

[19] Dershowitz, N., and Mitra, S. Path orderings for termination of associative-commutative rewriting. In *Conditional Term Rewriting Systems, proceedings third international workshop CTRS-92* (1993), M. Rusinowitch and J. Rémy, Eds., vol. 656 of *Lecture Notes in Computer Science*, Springer, pp. 168–174.

[20] Dick, J., Kalmus, J., and Martin, U. Automating the Knuth Bendix ordering. *Acta Informatica 28* (1990), 95–119.

[21] Drosten, K. *Termersetzungssysteme*, vol. 210 of *Informatik-Fachberichte*. Springer, 1989.

[22] Ferreira, M. C. F. *Termination of Term Rewriting: Well-foundedness, Totality and Transformations*. PhD thesis, Utrecht University, 1995.

[23] Ferreira, M. C. F. Dummy elimination in equational rewriting. In *Proceedings of the 7th Conference on Rewriting Techniques and Applications* (1996), H. Ganzinger, Ed., vol. 1103 of *Lecture Notes in Computer Science*, Springer, pp. 63–77.

[24] Ferreira, M. C. F., and Zantema, H. Syntactical analysis of total termination. In *Proceedings of the 4th International Conference on Algebraic and Logic Programming* (1994), G. Levi and M. Rodríguez-Artalejo, Eds., vol. 850 of *Lecture Notes in Computer Science*, Springer, pp. 204–222.

[25] Ferreira, M. C. F., and Zantema, H. Dummy elimination: Making termination easier. In *Fundamentals of Computation Theory, proceedings 10th international conference FCT95* (1995), H. Reichel, Ed., vol. 965 of *Lecture Notes in Computer Science*, Springer, pp. 243–252. Extended version appeared as report UU-CS-1994-47, Utrecht University, October 1994.

[26] Ferreira, M. C. F., and Zantema, H. Total termination of term rewriting. *Applicable Algebra in Engineering, Communication and Computing 7*, 2 (1996), 133–162. (Preliminary version appeared in Proceedings of RTA93, Lecture Notes in Computer Science 690, Springer, 1993.).

[27] Fokkink, W. J., and Zantema, H. Basic process algebra with iteration: Completeness of its equational axioms. *The Computer Journal 37*, 4 (1994), 259–267.

[28] Fokkink, W. J., and Zantema, H. Termination modulo equations by abstract commutation with an application to iteration. *Theoretical Computer Science 177* (1997), 407–423.

[29] Geser, A. *Relative termination.* PhD thesis, Universität Passau, 1990.

[30] Geser, A., Middeldorp, A., Ohlebusch, E., and Zantema, H. Relative undecidability in term rewriting. In *Proceedings of the Conference of the European Association of Computer Science Logic (CSL96)* (1997), D. van Dalen, Ed., Lecture Notes in Computer Science, Springer Verlag.

[31] Geser, A., Middeldorp, A., Ohlebusch, E., and Zantema, H. Relative undecidability in the termination hierarchy of single rewrite rules. In *Proceedings Theory and Practice of Software Development (TAPSOFT97, CAAP/FASE)* (1997), M. Bidoit and M. Dauchet, Eds., vol. 1214 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 237–248.

[32] Hardin, T., and Laville, A. Proof of termination of the rewriting system SUBST on CCL. *Theoretical Computer Science 46* (1986), 305–312.

[33] Hofbauer, D. Termination proofs by multiset path orderings imply primitive recursive derivation lengths. *Theoretical Computer Science 105*, 1 (1992), 129–140.

[34] Hofbauer, D., and Lautemann, C. Termination proofs and the length of derivations (preliminary version). In *Proceedings of the 3rd Conference on Rewriting Techniques and Applications* (1989), N. Dershowitz, Ed., vol. 355 of *Lecture Notes in Computer Science*, Springer, pp. 167–177.

[35] Huet, G., and Lankford, D. S. On the uniform halting problem for term rewriting systems. Rapport Laboria 283, INRIA, 1978.

[36] Jones, J. Universal diophantine equation. *Journal of Symbolic Logic 47* (1982), 549–571.

[37] Jouannaud, J.-P., Lescanne, P., and Reinig, F. Recursive decomposition order. In *IFIP Working Conference on Formal Description of Programming Concepts II.* North-Holland Publishing Company, 1982, pp. 331–348.

[38] Kamin, S., and Lévy, J. J. Two generalizations of the recursive path ordering. University of Illinois, 1980.

[39] Kaplan, S. Simplifying conditional term rewriting systems: unification, termination and confluence. *Journal of Symbolic Computation 4*, 3 (1987), 295–334.

[40] Kapur, D., Narendran, P., and Sivakumar, G. A path ordering for proving termination of term rewriting systems. In *Proceedings of the 10th Colloquium on Trees in Algebra and Programming (CAAP)* (1985), vol. 185 of *Lecture Notes in Computer Science*, Springer, pp. 173–187.

[41] Klop, J. W. Term rewriting systems: a tutorial. *Bulletin of the EATCS 32* (1987), 143–183.

[42] Klop, J. W. Term rewriting systems. In *Handbook of Logic in Computer Science*, D. G. S. Abramski and T. Maibaum, Eds., vol. 1. Oxford University Press, 1991.

[43] Knuth, D., and Bendix, P. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, J. Leech, Ed. Pergamon Press, 1970, pp. 263–297.

[44] KRUSKAL, J. Well-quasi-ordering, the tree theorem, and Vazsonyi's conjecture. *Trans. American Mathematical Society 95* (1960), 210–225.

[45] KURIHARA, M., AND OHUCHI, A. Modularity of simple termination of term rewriting systems. *Journal of IPS Japan 31*, 5 (1990), 633–642.

[46] LANKFORD, D. S. On proving term rewriting systems are noetherian. Tech. Rep. MTP–3, Louisiana Technical University, Ruston, 1979.

[47] LESCANNE, P. Termination of rewrite systems by elementary interpretations. In *Algebraic and Logic Programming* (1992), H. Kirchner and G. Levi, Eds., vol. 632 of *Lecture Notes in Computer Science*, Springer, pp. 21 – 36.

[48] LESCANNE, P. On termination of one rule rewrite systems. *Theoretical Computer Science 132* (1994), 395–401.

[49] MIDDELDORP, A. *Modular Properties of Term Rewriting Systems.* PhD thesis, Free University Amsterdam, 1990.

[50] MIDDELDORP, A., AND GRAMLICH, B. Simple termination is difficult. *Applicable Algebra in Engineering, Communication and Computing 6*, 2 (1995), 115–128. (Preliminary version appeared in Proceedings of RTA93, Lecture Notes in Computer Science 690, Springer, 1993.).

[51] MIDDELDORP, A., OHSAKI, H., AND ZANTEMA, H. Transforming termination by self-labelling. In *Proceedings of the 13th International Conference on Automated Deduction (CADE)* (1996), M. McRobbie and J. Slaney, Eds., vol. 1104 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 373–387.

[52] MIDDELDORP, A., AND ZANTEMA, H. Simple termination revisited. In *Proceedings of the 12th International Conference on Automated Deduction (CADE12)* (1994), A. Bundy, Ed., vol. 814 of *Lecture Notes in Computer Science*, Springer, pp. 451–465.

[53] MIDDELDORP, A., AND ZANTEMA, H. Simple termination of rewrite systems. *Theoretical Computer Science 175* (1997), 127–158.

[54] OHLEBUSCH, E. A note on simple termination of infinite term rewriting systems. Tech. Rep. 7, Universität Bielefeld, 1992.

[55] PLAISTED, D. The undecidability of self-embedding for term rewriting systems. *Information Processing Letters 20* (1985), 61–64.

[56] ROZENBERG, G., AND SALOMAA, A. *Cornerstones of Undecidability.* Prentice Hall, 1994.

[57] RUBIO, A., AND NIEUWENHUIS, R. A precedence-based total AC-compatible order. In *Proceedings of the 5th Conference on Rewriting Techniques and Applications* (1993), C. Kirchner, Ed., vol. 690 of *Lecture Notes in Computer Science*, Springer, pp. 374–388.

[58] RUSINOWITCH, M. On termination of the direct sum of term rewriting systems. *Information Processing Letters 26* (1987), 65–70.

[59] STEINBACH, J. Extensions and comparison of simplification orderings. In *Proceedings of the 3rd Conference on Rewriting Techniques and Applications* (1989), N. Dershowitz, Ed., vol. 355 of *Lecture Notes in Computer Science*, Springer, pp. 434–448.

[60] STEINBACH, J. Automatic termination proofs with transformation orderings. In *Proceedings of the 6th Conference on Rewriting Techniques and Applications* (1995), J. Hsiang, Ed., vol. 914 of *Lecture Notes in Computer Science*, Springer, pp. 11–25.

[61] STEINBACH, J. Simplification orderings: History of results. *Fundamenta Informaticae 24* (1995), 47–87.

[62] TOYAMA, Y. Counterexamples to termination for the direct sum of term rewriting systems. *Information Processing Letters 25* (1987), 141–143.

[63] TOYAMA, Y. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM 34*, 1 (1987), 128–143.

[64] ZANTEMA, H. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation 17* (1994), 23–50.

[65] ZANTEMA, H. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae 24* (1995), 89–105.

[66] ZANTEMA, H. Total termination of term rewriting is undecidable. *Journal of Symbolic Computation 20* (1995), 43–60.

[67] ZANTEMA, H. Termination of context-sensitive rewriting. In *Proceedings of the 8th International Conference on Rewriting Techniques and Applications* (1997), H. Comon, Ed., vol. 1232 of *Lecture Notes in Computer Science*, Springer, pp. 172–186.

[68] ZANTEMA, H., AND GESER, A. A complete characterization of termination of $0^p1^q \to 1^r0^s$. In *Proceedings of the 6th Conference on Rewriting Techniques and Applications* (1995), J. Hsiang, Ed., vol. 914 of *Lecture Notes in Computer Science*, Springer, pp. 41–55. Appeared as report UU-CS-1994-44, Utrecht University.

[69] ZANTEMA, H., AND GESER, A. Non-looping rewriting. Tech. Rep. UU-CS-1996-03, Utrecht University, January 1996. Available via `http://www.cs.ruu.nl/docs/research/publication/TechList2.html`.

Index