Automation of Recursive Path Ordering for Infinite Labelled Rewrite Systems

Adam Koprowski and Hans Zantema

Eindhoven University of Technology
Department of Computer Science
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
{A.Koprowski, H.Zantema}@tue.nl

Abstract. Semantic labelling is a transformational technique for proving termination of Term Rewriting Systems (TRSs). Only its variant with finite sets of labels was used so far in tools for automatic termination proving and variants with infinite sets of labels were considered not to be suitable for automation. We show that such automation can be achieved for semantic labelling with natural numbers, in combination with recursive path ordering (RPO). In order to do so we developed algorithms to deal with recursive path ordering for these infinite labelled systems. Using these techniques TPA, a tool developed by the first author, is the only current tool that can prove termination of the SUBST system automatically.

1 Introduction

Semantic labelling ([12]) is a well-known transformational termination technique for TRSs. The core of the idea is to interpret the function symbols in some model and use this interpretation to label the system. After this labelling all function symbols are equipped with a label and the resulting TRS is terminating if and only if the original one is terminating. This approach is often successful in the sense that proving termination for the labelled system is easier than for the original one. Typically, non simply terminating systems are often transformed to systems for which termination is easily proved by polynomials or recursive path ordering (RPO).

In recent years research in this area has focussed towards the automation of the termination proving process. Any termination technique, to be regarded successful, apart from being widely applicable needs to be suitable for automation. The best evidence of that is the annual termination competition ([1]) where termination tools written by different authors compete on a set of termination problems.

Automation of semantic labelling can be done straightforwardly if the models contain only finitely many elements, or in particular only 2 elements. For instance, this technique is used by the tools AProVE [6], TORPA [14] and TeParLa. However if we consider labelling with infinite sets of labels, like natural numbers, some complications show up as then the labelled system has an infinite

signature and contains infinitely many rules. For that reason such variants of semantic labelling were regarded as not feasible for automation and were not used in termination tools before 2005.

The main claim of this paper is that semantic labelling with a model consisting of the infinite set of natural numbers can be automated in combination with extending RPO implementation to the corresponding infinite signatures. By doing so, in the setting we are about to describe, one gets a widely applicable new termination technique. We give examples where this technique easily yields a termination proof whereas all other known techniques fail.

Apart from developing the theory we also implemented this technique in TPA[1] (Termination Proved Automatically), a termination tool developed by the first author, for which semantic labelling with natural numbers in combination with RPO is one of the main technique.

Before presenting the details of this technique, we give a motivating example.

Example 1. Consider the following TRS:

$$
\begin{array}{ll}
(1) & \lambda(x) \circ y \to \lambda(x \circ (1 \star (y \circ \uparrow))) \\
(2) & (x \star y) \circ z \to (x \circ z) \star (y \circ z) \\
(3) & (x \circ y) \circ z \to x \circ (y \circ z) \\
(4) & \mathsf{id} \circ x \to x \\
(5) & 1 \circ \mathsf{id} \to 1 \\
(6) & \uparrow \circ \mathsf{id} \to \uparrow \\
(7) & 1 \circ (x \star y) \to x \\
(8) & \uparrow \circ (x \star y) \to y
\end{array}
$$

This system, named δ_0 in [4] and essentially equivalent to system SUBST in [7] describes the process of substitution in combinatory categorical logic with 'λ' corresponding to currying, '$\circ$' to composition, 'id' to identity, '$\star$' to pairing and '1' and '$\uparrow$' to projections.

Termination of this system (implying termination of the process of explicit substitution in un-typed λ-calculus) is non-trivial and was the main result of [4] and [7]. However in [12, 13] a very simple proof was given using only semantic labelling with natural numbers followed by an application of RPO on the transformed system. Ability to reproduce this proof completely automatically was a first goal of this work, while a next goal was to make this approach fruitful in general. Both goals have been achieved.

This TRS will be a running example in this paper.

The outline of this paper is as follows. In Section 2 we present the required preliminaries on RPO. We continue in Section 3 by briefly introducing the technique of semantic labelling with natural numbers. Section 4 concentrates on adopting the recursive path ordering (RPO) to deal with infinite labelled systems. In section 5 we present practical evaluation of this technique along with two examples for which it is particularly suitable. We conclude in Section 6.

[1] For more information about TPA see `http://www.win.tue.nl/tpa`.

2 Recursive Path Ordering

Recursive path ordering (RPO) is an ordering introduced by Dershowitz [5] for proving termination of TRSs. We will briefly present it here. For a general introduction to term rewriting we refer to [2].

Let Σ be any signature, possibly infinite. Let $\succ$ be a well-founded order on Σ, called a precedence. Let τ be a status function on Σ, i.e., for every $f \in \Sigma$ the status $\tau(f)$ describes how to compare sequences of arguments of f: by multiset ordering or by lexicographic comparison in some direction. Then the corresponding recursive path ordering $\succ_{RPO}$ is defined as:

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

The main property of this ordering is the following:

> If $\ell \succ_{RPO} r$ for every rule $\ell \to r$ of a TRS R, then R is terminating.

So for proving termination by RPO one has to find a well-founded precedence $\succ$ and a status function τ such that $\ell \succ_{RPO} r$ for every rule $\ell \to r$. In tools this is typically done by collecting constraints on $\succ$ and checking whether these constraints give rise to a well-founded precedence $\succ$. In searching for these collections of constraints often choices are possible, also choices on τ, giving rise to back-tracking. The crucial algorithm required in this back-tracking procedure is to check whether a set of constraints on $\succ$ gives rise to a well-founded precedence $\succ$. For finite signatures this coincides with checking whether the corresponding graph is acyclic. For the infinite signatures, as we consider in this paper, it is more involved, but the basic frame of the algorithm remains the same. We will concentrate on the question of how to construct a well-founded precedence out of a collection of constraints on such a precedence.

3 Semantic Labelling with Natural Numbers

First we recall the main theory of semantic labelling.

A Σ-algebra (A, Σ_A) is defined to be a non-empty set A together with a map $[f] : A^n \to A$ for every $f \in \Sigma$, where n is the arity of f. Let $\mathcal{V}$ be a set of variables. For $\alpha : \mathcal{V} \to A$ we inductively define $[x, \alpha] = \alpha(x)$ for $x \in \mathcal{V}$, and $[f(t_1, \ldots, t_n), \alpha] = [f]([t_1, \alpha], \ldots, [t_n, \alpha])$.

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

In case the partial order $\geq$ coincides with equality then weak monotonicity holds for every operation, and the only requirement is $[\ell, \alpha] = [r, \alpha]$ for all rules $\ell \to r$ in R and all $\alpha : \mathcal{V} \to A$. In this case the quasi-model is called a *model*.

For labelling here we do not introduce an arbitrary labelling function as in [12, 13], but choose the particular case where this labelling function is the identity. This means that given a quasi-model $(A, \Sigma_A, \geq)$ for a TRS R over Σ, the labelled TRS $\overline{R}$ is defined as follows. The signature $\overline{\Sigma}$ consists of n-ary symbols $f_{a_1,\ldots,a_n}$, where f is an n-ary symbol from Σ and $a_1, \ldots, a_n \in A$. Given $\alpha : \mathcal{V} \to A$, the labelling function lab is defined inductively by

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

for $x \in \mathcal{V}$ and $f \in \Sigma$. Now $\overline{R}$ is defined to consist of the rules

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

for all $\alpha : \mathcal{V} \to A$ and all rules $\ell \to r$ of R. The TRS Decr is defined to consist of the rules

$$f_{a_1,\ldots,a_n}(x_1, \ldots, x_n) \to f_{b_1,\ldots,b_n}(x_1, \ldots, x_n)$$

for all $f \in \Sigma$ and $a_1, \ldots, a_n, b_1, \ldots, b_n$ satisfying $a_i > b_i$ for some i and $a_j = b_j$ for all $j \neq i$. Here $>$ denotes the strict part of $\geq$. Note that Decr is empty in the model case, i.e., if $\geq$ coincides with equality.
The main property of semantic labelling is the following:

R is terminating if and only if $\overline{R} \cup \mathsf{Decr}$ is terminating.

In this paper we focus on the case where $A = \mathbb{N}$, the natural numbers. The approach is as follows: for a TRS R for which we want to prove termination, search for interpretations in A such that $(A, \Sigma_A, \geq)$ is a quasi-model, and next try to prove termination of the infinite TRS $\overline{R} \cup \mathsf{Decr}$ by means of RPO. If this succeeds, then according to the main property of semantic labelling we have proved termination of R. In case $(A, \Sigma_A, \geq)$ happens to be a model, i.e., for all rules we have equality, then we choose $\geq$ on $A = \mathbb{N}$ to be equality, by which Decr is empty. In the other case we choose $\geq$ to be the usual order on $\mathbb{N}$. In this case Decr is not empty, but we may and shall restrict Decr to all rules of the shape

$$f_{a_1,\ldots,a_n}(x_1, \ldots, x_n) \to f_{b_1,\ldots,b_n}(x_1, \ldots, x_n)$$

for $f \in \Sigma$ and $a_1, \ldots, a_n, b_1, \ldots, b_n$ satisfying $a_i = b_i + 1$ for some i and $a_j = b_j$ for all $j \neq i$. This is valid since if $a_i > b_i$ then a_i can be obtained from b_i by taking the successor a number of times, so the rewrite relation $\to^+_{\mathsf{Decr}}$ is not changed by this modification of Decr. In the sequel we refer to $\overline{R} \cup \mathsf{Decr}$ as the *labelled system*.

Example 2. As an example, we apply this approach to the TRS R from Example 1 and the following interpretation in $\mathbb{N}$:

$$\begin{aligned} [\lambda](x) &= x + 1 & [1] &= 0 \\ [\star](x, y) &= \mathsf{max}(x, y) & [\uparrow] &= 0 \\ [\circ](x, y) &= x + y & [\mathsf{id}] &= 0 \end{aligned}$$

This interpretation is a quasi-model and after application of semantic labelling we obtain $\overline{R}$ consisting of the rules

$$
\begin{array}{lrll}
(1) & \lambda_i(x) \circ_{i+1,j} y \to \lambda_{i+j}(x \circ_{i,j} (1 \star_{0,j} (y \circ_{j,0} \uparrow))) & \\
(2a) & (x \star_{i,j} y) \circ_{i,k} z \to (x \circ_{i,k} z) \star_{i+k,j+k} (y \circ_{j,k} z) & \text{for } i \geq j \\
(2b) & (x \star_{i,j} y) \circ_{j,k} z \to (x \circ_{i,k} z) \star_{i+k,j+k} (y \circ_{j,k} z) & \text{for } i < j \\
(3) & (x \circ_{i,j} y) \circ_{i+j,k} z \to x \circ_{i,j+k} (y \circ_{j,k} z) & \\
(4) & \mathsf{id} \circ_{0,i} x \to x & \\
(5) & 1 \circ_{0,0} \mathsf{id} \to 1 & \\
(6) & \uparrow \circ_{0,0} \mathsf{id} \to \uparrow & \\
(7a) & 1 \circ_{0,i} (x \star_{i,j} y) \to x & \text{for } i \geq j \\
(7b) & 1 \circ_{0,j} (x \star_{i,j} y) \to x & \text{for } i < j \\
(8a) & \uparrow \circ_{0,i} (x \star_{i,j} y) \to y & \text{for } i \geq j \\
(8b) & \uparrow \circ_{0,j} (x \star_{i,j} y) \to y & \text{for } i < j
\end{array}
$$

and Decr consisting of the rules

$$
\begin{array}{ll}
(D_1) & \lambda_{i+1}(x) \to \lambda_i(x) \\
(D_{2a}) & x \circ_{i+1,j} y \to x \circ_{i,j} y \\
(D_{2b}) & x \circ_{i,j+1} y \to x \circ_{i,j} y \\
(D_{3a}) & x \star_{i+1,j} y \to x \star_{i,j} y \\
(D_{3b}) & x \star_{i,j+1} y \to x \star_{i,j} y.
\end{array}
$$

Here variables i, j, k run over $\mathbb{N}$.

The goal now is to represent such an infinite labelled system $\overline{R} \cup \mathsf{Decr}$ in such a way that we can search systematically for a suitable RPO proving its termination. Before doing so first we say something about the search for (quasi-)models.

As long as all basic interpretations are polynomials, checking whether this interpretation is a model coincides with checking whether $[\ell, \alpha] = [r, \alpha]$ for all rules $\ell \to r$ and all α. This is simply checking whether polynomials are equal. Checking whether an interpretation is a quasi-model coincides with checking whether $[\ell, \alpha] \geq [r, \alpha]$; this can be done along the lines of the standard way of checking for polynomial interpretations as described in [3, 9].

In TPA as a initial step symbols of arity > 2 are transformed to a number of binary symbols, so no symbols of arity > 2 occur anymore. Then in the basic setting the functions used as interpretations for constant, unary and binary symbols, respectively, are as follows[2]:

$$
\begin{array}{c}
\{0,\ 1\} \\
\{\lambda x.0,\ \lambda x.1,\ \lambda x.x,\ \lambda x.x+1,\ \lambda x.\max(0, x-1),\ \lambda x.2x,\ \lambda x.7x\} \\
\{\lambda x\, y.0,\ \lambda x\, y.1,\ \lambda x\, y.x+y,\ \lambda x\, y.x+y+3,\ \lambda x\, y.xy, \\
\lambda x\, y.x,\ \lambda x\, y.y,\ \lambda x\, y.\max(0, x-y),\ \lambda x\, y.\max(x, y),\ \lambda x\, y.\min(x, y)\}
\end{array}
$$

[2] Note that, for technical reasons, TPA actually uses $A = \mathbb{N} \setminus \{0, 1\}$ not $A = \mathbb{N}$ hence the actual functions being used are slight variants of those presented here.

So we may also want to use non-polynomial functions like min or max. Checking whether the required (in-)equalities hold is accomplished by first removing min and max functions by simple case analysis and then using the standard approach for polynomials.

Note however that while doing this case analysis we introduce side conditions, just like in Example 2. For F being a polynomial over $\mathbb{N}$ in n variables $a_1, \ldots, a_n$ let us abbreviate $\forall a_1, \ldots, a_n \in \mathbb{N}.F(a_1, \ldots, a_n) > 0$ by $F > 0$. So now the problem of comparing polynomials is not to check whether $F > 0$, as in the standard setting, but to check whether $\{F_i \geq 0\}_i \implies F > 0$ where the premise is a set of side conditions introduced by case analysis. This problem is undecidable as it is a generalization of polynomial comparison which is already undecidable. TPA uses a very simple and naive approximation of this problem and concludes $\{F_i \geq 0\}_i \implies F > 0$ only if $F > 0 \vee \exists i \in \{1, \ldots, n\}.F - F_i > 0$.

For instance comparison of function symbols $\circ_{i,k}$ and $\circ_{j,k}$ in rule $(2a)$ may require comparing polynomials $i + k$ and $j + k$ if $\phi_\circ = +$. For that we may use the side condition of this rule, $i \geq j$. We cannot conclude $i + k \geq j + k$ in general but by using side condition and subtracting i from the left hand side of this inequality and j from the right hand side we get $i + k - i \geq j + k - j$ which is trivially satisfied.

4 RPO for Infinite Labelled Systems

In this section we describe how RPO can be adapted to deal with labelled systems, more precisely, for infinite systems over infinite signatures obtained by labelling with natural numbers. In fact we do not change the definition of RPO, but we restrict the search space for possible precedences on the labelled symbols in such a way that this search can be automated and we have algorithms checking whether constraints on the precedence give rise to a well-founded precedence or not.

First in 4.1 we present theoretical foundations of those results and then in 4.2 we discuss the algorithmic approach for searching for a precedence satisfying given set of constraints.

4.1 Well-foundedness of a Precedence

The final precedence $\succ$ we search for will be of the following shape:

Definition 1 (Precedence description). *A* precedence description *consists of:*

- *for every $f \in \Sigma$ of arity n, $\phi_f : \mathbb{N}^n \to \mathbb{N}$ (we will call those functions* label synthesis functions*) and*
- $\mathsf{pd} : \Sigma \times \Sigma \to \{\bot, >, \geq, \top\}$.

These ingredients give rise to the following relation $\succ$:

$$\begin{aligned} f_{k_1,\ldots,k_n} \succ g_{l_1,\ldots,l_m} \iff\ & \mathsf{pd}(f,g) = \top \vee \\ & (\mathsf{pd}(f,g) = \geq \wedge \phi_f(k_1,\ldots,k_n) \geq \phi_g(l_1,\ldots,l_m)) \vee \\ & (\mathsf{pd}(f,g) = > \wedge \phi_f(k_1,\ldots,k_n) > \phi_g(l_1,\ldots,l_m)). \end{aligned}$$

So $\mathsf{pd}(f,g)$ indicates when we can conclude $f_{k_1,\ldots,k_n} \succ g_{l_1,\ldots,l_m}$ with: $\bot$ indicating that this can never be the case; $\top$ that it is always the case regardless of the labels of f and g; and $\geq$ and $>$ let us conclude $f_{k_1,\ldots,k_n} \succ g_{l_1,\ldots,l_m}$ if $\phi_f(k_1,\ldots,k_n)$ is, respectively, greater equal/strictly greater than $\phi_g(l_1,\ldots,l_m)$.

Typically this relation $\succ$ will not be an ordering as it may not be transitive but then it may be replaced by its transitive closure so by abuse of terminology we will call it a precedence.

We need criteria under which in the above setting we can conclude well-foundedness of $\succ$. If these criteria hold then termination of the labelled system, and hence of the original TRS, can be concluded if $\ell \succ_{RPO} r$ for all rules $\ell \to r$ in the labelled system. So our approach can be summarized as follows: collect constraints on pd and the label synthesis functions ϕ_f from the requirement that $\ell \succ_{RPO} r$ for all rules $\ell \to r$, and then check whether this gives rise to a well-founded precedence $\succ$.

This well-foundedness criterion is captured by the following theorem. A function $\mathsf{pd} : \Sigma \times \Sigma \to \{\bot, >, \geq, \top\}$ gives rise to a *precedence graph* having Σ as its node set, and having three kinds of directed edges:

- an *unconditional* edge from f to g if $\mathsf{pd}(f,g) = \top$, denoted by a double arrow $\Longrightarrow$;
- a *strict* edge from f to g if $\mathsf{pd}(f,g) = >$, denoted by a single arrow $\longrightarrow$;
- a *non-strict* edge from f to g if $\mathsf{pd}(f,g) = \geq$, denoted by a dotted arrow $\cdots\!\!\succ$;

Theorem 1 (Well-foundedness of a precedence). *In the above setting a precedence description* pd *gives rise to a well-founded precedence* $\succ$ *if every cycle in the corresponding precedence graph*

(1) contains no unconditional edge, and
(2) contains at least one strict edge.

Proof. Suppose that the precedence is not well-founded. This means that there is an infinite sequence $f_{k_1,\ldots,k_n} \succ g_{l_1,\ldots,l_m} \succ \ldots$. Every step in this reduction corresponds to an edge in the precedence graph. Since this sequence is infinite it must traverse some cycle in the precedence graph (which is finite) infinitely often. Every cycle contains only strict and non-strict edges due to (1), which gives rise to the inequalities on ϕ functions as depicted below.

$$\cdots \longrightarrow f \longrightarrow g \longrightarrow \cdots$$

$$\ldots \geq \ \phi_f(k_1,\ldots,k_n) \quad \geq \quad \phi_g(l_1,\ldots,l_m) \ \geq \ldots$$

Due to (2) at least one of those inequalities is strict which gives rise to a decreasing weight along a cycle. Hence no cycle can be traversed infinitely often. Contradiction, we conclude well-foundedness of $\succ$. □

To make the approach feasible, but still applicable to interesting examples, it is natural to restrict the choice for the label synthesis functions. In TPA the

choice has been made to choose ϕ to always be identity for unary symbols. For binary symbols ϕ is chosen to be one of the three functions: summation $+$, left projection π_1 and right projection π_2. This set of synthesis functions may seem quite restricted but it works reasonably well in practice whereas a bigger set would lead to a bigger search space.

One of the constraints implied by the rule

$$\lambda_i(x) \circ_{i+1,j} y \;\rightarrow\; \lambda_{i+j}(x \circ_{i,j} (1 \star_{0,j} (y \circ_{j,0} \uparrow)))$$

where i, j runs over the naturals, will be $\circ_{i+1,j} \succ \lambda_{i+j}$, for all i, j. Since ϕ_λ is fixed to be the identity, according to the definition of $\succ$ this gives three possibilities:

1. $\mathsf{pd}(\circ, \lambda) \;=\; \top$,
2. $\mathsf{pd}(\circ, \lambda) \;= \geq \;\wedge\; \phi_\circ(i+1, j) \geq i + j$ for all i, j,
3. $\mathsf{pd}(\circ, \lambda) \;= > \;\wedge\; \phi_\circ(i+1, j) > i + j$ for all i, j.

By re-using the algorithm for comparing polynomials it can be determined that among the three possibilities for $\phi_\circ$ cases 2 and 3 only hold if $\phi_\circ = +$, from which case 3 is preferred since by Theorem 1 strict edges are preferred over non-strict edges.

4.2 Algorithm for Computing a Well-founded Precedence

In general for every pair (f, g) the RPO constraints will give rise to a list of cases. Each case consists of a choice for $\mathsf{pd}(f, g)$ being $\top$, $\geq$ or $>$, and in case this is not $\top$, also a choice for ϕ_f and ϕ_g. Now the question is whether for every pair (f, g) a choice can be made in such a way that conditions of Theorem 1 are satisfied so that this choice gives rise to a well-founded precedence. Note that for every $f \in \Sigma$ in a precedence there is a single, global function ϕ_f corresponding to it. However in the search procedure we do not know what this ϕ_f should be and hence during the search we allow using different functions for comparison with different symbols and only at the end we conclude which one should be chosen for every function symbol. We will express all those choices for ϕ and pd, in, what we call, a precedence description scheme.

Definition 2 (Precedence description scheme).
We define a precedence description scheme *as a function from pairs of function symbols to a set of possible implementations of ordering description on these function symbols which is either $\bot$ or $\top$ or one of $>$, $\geq$ accompanied with ϕ functions to be used for comparing those two symbols.*

$$\mathsf{pds}(f, g) \subseteq \{\bot\} \cup (\{>, \geq\} \times \mathbb{N}^{\mathbb{N}^n} \times \mathbb{N}^{\mathbb{N}^m}) \cup \{\top\}$$

where n is the arity of f and m arity of g.

Now we will say that a precedence description $(\{\phi_f\}_{f \in \Sigma}, \mathsf{pd})$ is compatible with a precedence description scheme pds *if:*

$$\forall f, g \in \Sigma \;.\; \begin{cases} \mathsf{pd}(f, g) = \top \implies \top \in \mathsf{pds}(f, g) \\ \mathsf{pd}(f, g) = \geq \implies (\geq, \phi_f, \phi_g) \in \mathsf{pds}(f, g) \\ \mathsf{pd}(f, g) = > \implies (>, \phi_f, \phi_g) \in \mathsf{pds}(f, g) \end{cases}$$

Let us illustrate the notion of precedence description scheme on an example.

Example 3. Consider the labelled system from Example 2. A possible branching of an application of RPO to that system gives the following sets of constraints for respective rules; note that for $\circ$ a lexicographic left-to-right status is essential.

$$\begin{array}{ll}
(1) & \{\circ_{i+1,j} > \lambda_{i+j},\ \circ_{i+1,j} > \circ_{i,j},\ \circ_{i+1,j} > \star_{0,j},\\
 & \quad \circ_{i+1,j} > 1,\ \circ_{i+1,j} > \circ_{j,0},\ \circ_{i+1,j} > \uparrow\}\\
(2a) & \{\circ_{i,k} > \star_{i+k,j+k},\ \circ_{i,k} \geq \circ_{j,k}\} \text{ with } i \geq j\\
(2b) & \{\circ_{j,k} > \star_{i+k,j+k},\ \circ_{j,k} \geq \circ_{i,k}\} \text{ with } i < j\\
(3) & \{\circ_{i+j,k} = \circ_{i,j+k},\ \circ_{i+j,k} > \circ_{j,k}\}\\
(D_1) & \{\lambda_{i+1} > \lambda_i\}\\
(D_{2a}) & \{\circ_{i+1,j} > \circ_{i,j}\}\\
(D_{2b}) & \{\circ_{i,j+1} > \circ_{i,j}\}\\
(D_{3a}) & \{\star_{i+1,j} > \star_{i,j}\}\\
(D_{3b}) & \{\star_{i,j+1} > \star_{i,j}\}
\end{array}$$

Now with a finite set of label synthesis function, transformation of those constrains to a precedence description scheme can be easily accomplished. For instance for $\mathsf{pds}(\circ, \star)$ we need to consider the following three constraints:

$$\circ_{i+1,j} > \star_{0,j} \qquad \circ_{i,k} > \star_{i+k,j+k} \text{ with } i \geq j \qquad \circ_{j,k} > \star_{i+k,j+k} \text{ with } i < j$$

Given a finite set of label synthesis functions we can consider all possible combinations of synthesis functions for $\circ$ and $\star$ and analyze the resulting polynomial constraints. In TPA the set of label synthesis functions for binary symbols consists of π_1, π_2 and $+$. If $\circ$ always bigger than $\star$ then we trivially have those inequalities, so $\top \in \mathsf{pds}(\circ, \star)$. For remaining conditional cases we easily observe that only $+$ is possible as a label synthesis function for $\circ$ whereas only projections are allowed for $\star$. We get $(\geq, +, \pi_1) \in \mathsf{pds}(\circ, \star)$ because $(i+1)+j \geq \pi_1(0, j)$, $i + k \geq \pi_1(i + k, j + k)$ and $i < j \implies j + k \geq \pi_1(i + k, j + k)$. Similarly $(\geq, +, \pi_2) \in \mathsf{pds}(\circ, \star)$. Continuing such analysis we end up with the following precedence description scheme. For all f and g for which $\mathsf{pds}(f, g)$ is not listed in the table below we have $\mathsf{pds}(f, g) = \{\bot\}$.

$$\begin{array}{ll}
\mathsf{pds}(\circ, 1) = \{\top\} & \mathsf{pds}(\circ, \lambda) = \{(>, +, \mathsf{id}), \top\}\\
\mathsf{pds}(\circ, \uparrow) = \{\top\} & \mathsf{pds}(\circ, \star) = \{(\geq, +, \pi_1), (\geq, +, \pi_2), \top\}\\
\mathsf{pds}(\circ, \circ) = \{(>, +, +)\} & \mathsf{pds}(\lambda, \lambda) = \{(>, \mathsf{id}, \mathsf{id})\}\\
\mathsf{pds}(\star, \star) = \{(>, +, +)\} &
\end{array}$$

To summarize our approach: given TRS find an interpretation that is a (quasi-)model and label the system. Now find a RPO termination proof for that system. This proof gives rise to a number of constraints on precedence that can be transformed to a precedence description scheme pds as in Example 3. Now our task is to find a precedence that satisfies those constraints and is well-founded. That is we are looking for a precedence description pd which is: (a) compatible with pds and (b) satisfies conditions of Theorem 1.

Before presenting an appropriate algorithm we first observe that a simpler problem of finding any precedence description compatible with a given precedence description scheme is NP-complete. We prove that by showing reduction from 3-coloring problem of graphs. Given undirected graph $G = (V, E)$ problem of *3-coloring of* G is to decide whether there exist:

$$f : V \to \{1, 2, 3\} \quad \text{such that} \quad \forall (u, v) \in E \quad f(u) \neq f(v)$$

This problem is well-known to be NP-complete.

Theorem 2. *Suppose at least three different ϕ functions are allowed. Then given a precedence description scheme* pds*, the problem of finding a precedence description compatible with it is NP-complete.*

Proof. Easy. See [11].

Before we present the algorithm let us put the problem in a more practical light by discussing it in the context in which it occurs in TPA. As mentioned before TPA uses precisely three different ϕ functions for binary symbols meaning that we are on the border of conditions posted in Theorem 2. If it had two the problem would correspond to 2-coloring which can be solved in polynomial time. But we believe that all three functions are important and we do not want to get rid of any of them. Moreover we are about to describe an algorithm that in practice performs very well and takes negligible time in the whole search procedure.

First let us observe that from the precedence description scheme pds we can already determine the structure of the precedence graph for a precedence description pd compatible with pds. Let us note that due to the construction of the precedence description scheme for any f and g we either have $\mathsf{pds}(f, g) = \{\bot\}$ or $\top \in \mathsf{pds}(f, g) \wedge \bot \notin \mathsf{pds}(f, g)$. Now if $\mathsf{pds}(f, g) = \{\bot\}$ then we can simply choose $\mathsf{pd}(f, g) = \bot$ and hence there is no edge from f to g in the precedence graph. Otherwise f and g are connected with an edge although at this point we do not know yet what is the type of this edge.

The key to get an efficient algorithm is the observation that we can detect strongly connected components (SCCs) in the precedence graph and treat them separately. Since precedence graphs are typically spare, meaning that SCCs are small, by doing so we increase the efficiency greatly. Note that all the edges between f and g belonging to different SCCs do not lie on any cycle and hence cannot violate conditions of the Theorem 1. Thus they can safely be changed to unconditional edges (as then $\top \in \mathsf{pds}(f, g)$). On the other hand there cannot be an unconditional edge connecting two nodes from the same SCC since all the edges within SCC belong to some cycle, so all such options can be dropped from the precedence description scheme.

So now we can localize further reasoning to a single SCC and we can limit to strict and non-strict edges only. We still need to find ϕ functions for all function symbols and the appropriate ordering of those symbols.

Firstly for all function symbols we compute their predecessors and successors in the precedence graph:

$$\mathsf{IN}_f = \{g \mid \mathsf{pds}(g,f) \neq \{\bot\},\ \text{g in the same SCC as f}\}$$
$$\mathsf{OUT}_f = \{g \mid \mathsf{pds}(f,g) \neq \{\bot\},\ \text{g in the same SCC as f}\}$$

Now we can compute possible label synthesis functions for every function symbol:

$$\mathsf{PSF}_f = \bigcap_{g \in \mathsf{OUT}_f} \{\phi_f \mid (\geq / >, \phi_f, \phi_g) \in \mathsf{pds}(f,g)\} \cap \bigcap_{g \in \mathsf{IN}_f} \{\phi_f \mid (\geq / >, \phi_g, \phi_f) \in \mathsf{pds}(g,f)\}$$

If for any f, $\mathsf{PSF}_f = \emptyset$ then we can finish with a negative answer. Otherwise we refine ps in the following way:

$$\mathsf{pds}'(f,g) := \{(\geq / >, \phi_f, \phi_g) \in \mathsf{pds}(f,g) \mid \phi_f \in \mathsf{PSF}_f,\ \phi_g \in \mathsf{PSF}_g\}$$

We continue this procedure as long as there are some changes in the refinement of pds. If at any point for any f and g, $\mathsf{pds}(f,g) = \emptyset$ we finish with negative answer. Hopefully we arrive at pds with all entries being singletons in which case we have only one potential solution; otherwise we need to consider all the possible choices. At the end we check whether condition (2) of Theorem 1 is satisfied that is whether there are no cycles containing non-strict edges only.

For a more detailed summary of the whole procedure we refer the reader to [11].

5 Practical evaluation

The technique of semantic labelling with natural numbers has been implemented in TPA. In Section 5.1 we discuss the role of this technique in TPA and in Section 5.2 we discuss two examples.

5.1 TPA - Termination Proved Automatically

The results presented below come from the evaluation of TPA on the database of 773 TRSs, which is used for the Termination Competition and is available at the following address: `http://www.lri.fr/~marche/tpdb`.

Semantic labelling with natural numbers is one of the techniques implemented in TPA. For a recent version of the tool out of 432 successful proofs in 85 this techniques has been applied. Moreover after semantic labelling with natural numbers has been switched off 32 of those systems could not be proven terminating anymore.

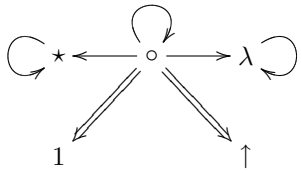
Another interesting experiment is the evaluation of the claim we made in Section 4.2, namely that typically SCCs in the precedence graph are small and

hence the algorithm is efficient. For 115 systems for which semantic labelling with natural numbers was applicable we calculated the size of the biggest SCC occurring in the analysis of that system (note that often many different labellings are tried resulting in many applications of the algorithm from Section 4.2). The average of those values was less than 5 confirming the claim that in practice SCCs are very small. Also the time spent on the execution of the algorithm in question on average summed up to less than 1% of the total running time of TPA.

5.2 Examples

In this section we would like to finish our analysis of the SUBST system introduced in Example 1 and also present one new example. We will show that using our approach both those systems can be easily proved terminating. Indeed TPA produces termination proof for them whereas, at the time being, no other termination tool can deal with those systems.

Example 4. Let us continue with Example 3, where we presented a precedence description scheme for this system. Below we depict the precedence graph corresponding to this scheme.



Using algorithm from Section 4.2 we first observe that all nodes are in separate SCCs. So first we replace all edges between different nodes by unconditional edges. Then we are left with no choice and we end up with the following precedence:

$$\begin{array}{ll} \mathsf{pd}(\circ,\lambda) = \top & \\ \mathsf{pd}(\circ,\star) = \top & \\ \mathsf{pd}(\circ,1) = \top & \phi_\circ = + \\ \mathsf{pd}(\circ,\uparrow) = \top & \phi_\lambda = \mathsf{id} \\ \mathsf{pd}(\circ,\circ) = > & \phi_\star = + \\ \mathsf{pd}(\lambda,\lambda) = > & \\ \mathsf{pd}(\star,\star) = > & \end{array}$$

which can be written down as:

$$\begin{array}{rl} \circ_{i,j} > \lambda_k & \text{for all } i,j,k \\ \circ_{i,j} > \star_{k,l} & \text{for all } i,j,k,l \\ \circ_{i,j} > 1 & \text{for all } i,j \\ \circ_{i,j} > \uparrow & \text{for all } i,j \\ \circ_{i,j} > \circ_{k,l} & \text{if } i+j > k+l \\ \lambda_i > \lambda_k & \text{if } i > k \\ \star_{i,j} > \star_{k,l} & \text{if } i+j > k+l \end{array}$$

One can easily check that all the rules of the SUBST TRS can be oriented using RPO with this precedence. This is also essentially the same solution as presented in [13].

In the termination competition it is allowed that tool authors submit up to 5 TRS, the so-called secret problems. Those problems are then merged into the main database and all the tools compete on them trying to prove their termination. All 5 systems submitted by TPA in the 2005 competition could be proven terminating using semantic labelling with natural numbers and RPO and for 4 of those systems TPA was the only tool that could prove their termination. Below we present one of those systems. Please note that this is a very natural TRS and not some artificial, cooked-up system.

Example 5. Consider the following TRS describing a GCD (*Greatest common divisor*) computation in a straightforward way.

$$\begin{aligned}
\mathsf{min}(x,0) &\to 0\\
\mathsf{min}(0,y) &\to 0\\
\mathsf{min}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{s}(\mathsf{min}(x,y))\\
\mathsf{max}(x,0) &\to x\\
\mathsf{max}(0,y) &\to y\\
\mathsf{max}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{s}(\mathsf{max}(x,y))\\
x-0 &\to x\\
\mathsf{s}(x)-\mathsf{s}(y) &\to x-y\\
\mathsf{gcd}(\mathsf{s}(x),0) &\to s(x)\\
\mathsf{gcd}(0,s(y)) &\to s(y)\\
\mathsf{gcd}(\mathsf{s}(x),\mathsf{s}(y)) &\to \mathsf{gcd}(\mathsf{max}(x,y)-\mathsf{min}(x,y),\mathsf{s}(\mathsf{min}(x,y)))
\end{aligned}$$

Consider the following interpretation of function symbols in $\mathbb{N}$:

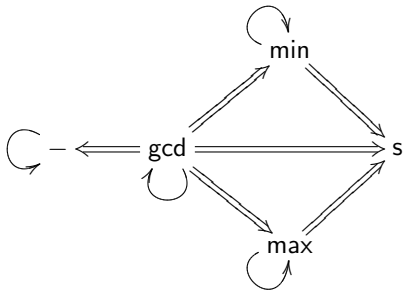
$$\begin{aligned}
[\mathsf{s}](x) &= x+1 & [0] &= 0\\
[\mathsf{min}](x,y) &= \mathsf{min}(x,y) & [\mathsf{max}](x,y) &= \mathsf{max}(x,y)\\
[-](x,y) &= x & [\mathsf{gcd}](x,y) &= x+y
\end{aligned}$$

This interpretation is a quasi-model and after application of semantic labelling it gives the following TRS:

$$\begin{aligned}
\mathsf{min}_{i,0}(x,0) &\to 0 && \\
\mathsf{min}_{0,j}(0,y) &\to 0 && \\
\mathsf{min}_{i+1,j+1}(\mathsf{s}_i(x),\mathsf{s}_j(y)) &\to \mathsf{s}_j(\mathsf{min}_{i,j}(x,y)) && \text{for } i \geq j\\
\mathsf{min}_{i+1,j+1}(\mathsf{s}_i(x),\mathsf{s}_j(y)) &\to \mathsf{s}_i(\mathsf{min}_{i,j}(x,y)) && \text{for } i < j\\
\mathsf{max}_{i,0}(x,0) &\to x && \\
\mathsf{max}_{0,j}(0,y) &\to y && \\
\mathsf{max}_{i+1,j+1}(\mathsf{s}_i(x),\mathsf{s}_j(y)) &\to \mathsf{s}_i(\mathsf{max}_{i,j}(x,y)) && \text{for } i \geq j\\
\mathsf{max}_{i+1,j+1}(\mathsf{s}_i(x),\mathsf{s}_j(y)) &\to \mathsf{s}_j(\mathsf{max}_{i,j}(x,y)) && \text{for } i < j\\
x -_{i,0} 0 &\to x &&
\end{aligned}$$

$$
\begin{aligned}
\mathsf{s}_i(x) -_{i+1,j+1} \mathsf{s}_j(y) &\to x -_{i,j} y \\
\mathsf{gcd}_{i+1,j+1}(\mathsf{s}_i(x), \mathsf{s}_j(y)) &\to \mathsf{gcd}_{i-j,j}(\mathsf{max}_{i,j}(x,y) -_{i,j} \mathsf{min}_{i,j}(x,y), \quad \text{for } i \geq j \\
&\qquad \mathsf{s}_j(\mathsf{min}_{i,j}(x,y))) \\
\mathsf{gcd}_{i+1,j+1}(\mathsf{s}_i(x), \mathsf{s}_j(y)) &\to \mathsf{gcd}_{j-i,i}(\mathsf{max}_{i,j}(x,y) -_{j,i} \mathsf{min}_{i,j}(x,y), \quad \text{for } i < j \\
&\qquad \mathsf{s}_j(\mathsf{min}_{i,j}(x,y)))
\end{aligned}
$$

Termination of the union of the above rules and decreasing rules can be proved with RPO. This time the description scheme leaves more choice for the label synthesis functions for different symbols but again all nodes are in separate SCCs and our algorithm easily yields the following precedence description:



$$
\begin{aligned}
\phi_{min} &= \pi_1 \\
\phi_{max} &= \pi_1 \\
\phi_s &= \mathsf{id} \\
\phi_- &= \mathsf{id} \\
\phi_{gcd} &= +
\end{aligned}
$$

which corresponds to the following well-founded precedence:

$$
\begin{array}{llll}
\mathsf{min}_{i,j} > \mathsf{min}_{k,l} & \text{if } i > k & \mathsf{gcd}_{i,j} > \mathsf{gcd}_{k,l} & \text{if } i+j > k+l \\
\mathsf{min}_{i,j} > \mathsf{s}_k & \text{for all } i,j,k & \mathsf{gcd}_{i,j} > \mathsf{min}_{k,l} & \text{for all } i,j,k,l \\
\mathsf{max}_{i,j} > \mathsf{max}_{k,l} & \text{if } i > k & \mathsf{gcd}_{i,j} > \mathsf{s}_k & \text{for all } i,j,k \\
\mathsf{max}_{i,j} > \mathsf{s}_k & \text{for all } i,j,k & \mathsf{gcd}_{i,j} > \mathsf{max}_{k,l} & \text{for all } i,j,k,l \\
-_{i,j} > -_{k,l} & \text{if } i > k & \mathsf{gcd}_{i,j} > -_{k,l} & \text{for all } i,j,k,l
\end{array}
$$

Another option to tackle this system would be to use predictive labelling [8]. Roughly speaking that would allow us not to give an interpretation for gcd symbol and ignore the rules defining gcd while checking the quasi-model conditions.

6 Conclusions and Further Research

In this paper we presented a way of automating RPO extended to infinite systems. This allows its use for systems that were transformed using semantic labelling with natural numbers. We explained how the combination of those two techniques can be employed for proving termination of rewriting and presented examples where it is successful whereas all other techniques seem to fail. Our description of automation makes it possible to use this technique in termination tools. Finally we briefly described the way in which it has been implemented and its evaluation in a termination tool TPA, developed by the first author.

Several extensions are possible. There is quite a lot of choice and questions that arise while employing this technique: what functions to use for interpretations? And for label synthesis functions? How to deal with case analysis and corresponding extended problem of comparing polynomials? Those questions can be studied further in order to make the most out of this technique. Another interesting issue is investigation of combination of other standard techniques, like for instance Knuth-Bendix order (KBO, [10]), with semantic labelling with natural numbers.

Acknowledgments

Authors would like to thank Gerhard Woeginger for his valuable ideas and remarks that contributed to the material presented in Section 4.2.

References

1. The termination competition. `http://www.lri.fr/~marche/termination-competition`.
2. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
3. A. B. Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Sci. Comput. Program.*, 9(2):137–159, 1987.
4. P.-L. Curien, T .Hardin, and A. Ríos. Strong normalization of substitutions. In Ivan M. Havel and Václav Koubek, editors, *MFCS*, volume 629 of *Lecture Notes in Computer Science*, pages 209–217. Springer, 1992.
5. N. Dershowitz. Orderings for term-rewriting systems. *Theor. Comput. Sci.*, 17:279–301, 1982.
6. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated termination proofs with AProVE. In Vincent van Oostrom, editor, *RTA*, volume 3091 of *Lecture Notes in Computer Science*, pages 210–220. Springer, 2004.
7. T. Hardin and A. Laville. Proof of termination of the rewriting system SUBST on CCL. *Theor. Comput. Sci.*, 46(2-3):305–312, 1986.
8. N. Hirokawa and A. Middeldorp. Predictive labeling. In Proceedings of *RTA*, LNCS, 2006. To appear.
9. H. Hong and D. Jakus. Testing Positiveness of Polynomials. *J. Autom. Reasoning*, 21(1): 23–38, 1998
10. D. Knuth and P. Bendix. Simple word problems in universal algebras. *Computational Problems in Abstract Algebra*, pages 263–297, 1970.
11. A. Koprowski and H. Zantema. Recursive Path Ordering for Infinite Labelled Rewrite Systems. Technical Report CS-Report 06-17, Eindhoven Univ. of Tech., April 2006.
12. H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24(1/2):89–105, 1995.
13. H. Zantema. Term Rewriting Systems. Volume 55 of *Cambridge Tracts in TCS*, chapter 6, pages 181–259. Cambridge University Press, 2003.
14. H. Zantema. TORPA: Termination of rewriting proved automatically. In Vincent van Oostrom, editor, *RTA*, volume 3091 of *Lecture Notes in Computer Science*, pages 95–104, 2004.