Termination of string rewriting proved automatically

Citation for published version (APA):

Document status and date:
Published: 01/01/2003

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaaccess@tue.nl
providing details and we will investigate your claim.
Termination of String Rewriting Proved Automatically

H. Zantema

Department of Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB, Eindhoven, The Netherlands
e-mail h.zantema@tue.nl

Abstract

In this paper it is described how a combination of semantic labelling, polynomial interpretations, recursive path order and the dependency pair method can be used for automatically proving termination of an extensive class of string rewriting systems (SRSs). The tool implementing these techniques is called TORPA: Termination of Rewriting Proved Automatically. All termination proofs generated by TORPA are easy to read and check, but for many of them finding a termination proof for the same SRS would be a hard job for a human. This paper contains all underlying theory, describes how search for a termination proof is implemented, and includes many examples.

1 Introduction

In the last few decades many techniques have been developed for proving termination of rewriting. In the last years work in this area concentrates on proving termination automatically: the development of tools by which a rewrite system can be entered and by which fully automatically a termination proof is generated. A strong impulse in the development of these tools was given by the contest on the International Workshop on Termination in Valencia in June 2003, which is planned to be the first in a regular series of contests.

In this paper we describe the tool TORPA: Termination of Rewriting Proved Automatically. This tool has been developed by the author. After having ideas in mind for years the actual implementation started in July 2003.

The present version only works on string rewriting. On the one hand this is a strong restriction compared to general term rewriting. On the other hand string rewriting is a natural and widely accepted paradigm with full computational power. For instance, Turing machine computation can be seen as a particular way of string rewriting. To illustrate the power of string rewriting, in Section 7 we will represent the well-known Collatz problem as the termination property of a surprisingly small SRS.
There are many small SRSs from a wide variety of origins for which termination is proved fully automatically by TORPA within a fraction of a second. For some of them all other techniques for automatically proving termination seem to fail, being a justification of writing this paper.

The main feature of TORPA is that an SRS is given as input and that TORPA generates a proof that this SRS is terminating. This proof is given in text. It is given in such a way that any human familiar with the basic techniques used in TORPA as they are described in this paper, can easily read and check the proof. The four basic techniques used are

- polynomial interpretations ([12]),
- recursive path order ([5]),
- semantic labelling ([15]), and
- dependency pairs ([1]).

Polynomial interpretations and recursive path order are complete techniques to prove termination, while semantic labelling and dependency pairs are techniques for transforming an SRS to another one in such a way that termination of the original SRS can be concluded from (relative) termination of the transformed SRS. Semantic labelling can be seen as the most important transformation used in TORPA.

TORPA is applicable for proving relative termination rather than termination. The property of relative termination of an SRS $R$ relative to an SRS $S$ means that any reduction with respect to $R \cup S$ contains at most finitely many $R$-steps. If $S = \emptyset$ this coincides with termination of $R$. The reason that relative termination is included is twofold:

- There are applications in which the problem to be solved can naturally be expressed as relative termination, and not as termination. In particular here we think of liveness problems.
- The technique of dependency pairs can be seen as a technique by which termination of a rewrite system is proved via proving relative termination of a transformed rewrite system. In this way in our implementation of the dependency pair method techniques for proving relative termination are used to prove plain termination of an SRS.

For all of the four basic techniques we do not use the full general version but only a particular case suitable for automation. On the other hand the techniques are extended to cover relative termination rather than termination. The power of TORPA is not in the techniques themselves (all of them are well-known for several years), but in the way various instances of them are combined yielding both power and efficiency.

In this paper we describe and motivate the choices of the instances of the four techniques, and give many examples of proofs as they are generated by TORPA. For each of the techniques we give a self-contained exposure of the soundness, except for recursive path order which plays only a minor role in TORPA. It turns out that for each of the other three techniques the proof of correctness can be given in no more than one page.
Other tools like TTT ([9]), CiME ([4]) and AProVE ([6]) combine dependency pairs and path orders too, and apply them much more involved then we do. The tool Termptation ([3]) is based on semantic path order. They turn out to be the best tools for proving termination of term rewriting of the moment. However, applied to small SRSs these tools appear to be much weaker than TORPA.

A completely different approach is chosen in the tool MatchBox [7]). By this tool match-boundedness of SRSs can be proved automatically, proving termination of the SRS. For many small SRSs this turns out to be a remarkably strong tool: it is able to prove termination automatically for many SRSs for which TORPA fails. However, there are many terminating SRSs for which match-boundedness does not hold, hence MatchBox fails, while termination is easily proved by TORPA.

Mainly due to the use of semantic labelling TORPA is able to prove (relative) termination of a great number of SRSs. It has been thought by several people that semantic labelling is unsuitable for automatically finding termination proofs since a (quasi-)model has to be chosen. However, by restricting to (quasi-)models consisting of only two elements it turns out that the number of candidates for (quasi-)models is not too big, and trying several of these candidates is a fruitful approach. This approach to semantic labelling can be seen as the core of TORPA. By absence of general heuristics for choosing a (quasi-)model the program often tries a number of randomly chosen attempts. It turns out that by trying typically 100 valid (quasi-)models this approach works surprisingly well. In the full program such a search in 100 randomly generated (quasi-)models is executed for many variants of the given SRS.

All four techniques can be applied to term rewriting rather than string rewriting. However, it is not always clear how our choices of using the techniques generalize to term rewriting maintaining the balance between power and tractability. Moreover, TORPA strongly uses the observation that an SRS is terminating if and only if its reverse is terminating. Here its reverse is the SRS obtained by reversing all its lhs’s and rhs’s of its rule. A similar transformation is not available for general term rewriting. Applying the ideas of TORPA to term rewriting is object of further research.

TORPA is freely available in two versions:

- A full version written in Delphi with a graphical user interface, including facilities for editing SRSs. The executable file of this version is available. This runs directly in a Windows environment without any installation. Detailed information about the input format is obtained by pushing the help button in this program.

- A plain version written in Pascal. Here the SRS should be given as the input and the text of the termination proof will be given as the output. The generated text should be exactly the same as in the first version, up to differences in the used random generator. It has the facility to apply the tool on a sequence of SRSs rather than only one. From this version the source code is available. This code is in standard format as accepted e.g. by Free Pascal and GNU Pascal, both being freely available for all standard platforms.
Both versions can be downloaded from

www.win.tue.nl/~hzantema/torpa.html

where also detailed information is given about the input format for the plain Pascal version. The present version is called version 1.1.

The structure of this paper is as follows. First we give preliminaries of string rewriting and relative termination. Then in four consecutive sections we discuss each of the four basic techniques. Next the Collatz problem is discussed as an example showing the power of string rewriting, applying techniques developed in this paper and even applying TORPA in a proof. In the final section we give conclusions and discuss further research. Throughout the paper various examples of TORPA output is given. To distinguish text generated by TORPA from the text of the paper the text generated by TORPA is always given in typewriter font.

2 Preliminaries

A string rewrite system (SRS) over an alphabet \( \Sigma \) is a set \( R \subseteq \Sigma^+ \times \Sigma^* \). Elements \((l, r) \in R\) are called rules and are written as \( l \rightarrow r \); \( l \) is called the left hand side (lhs) and \( r \) is called the right hand side (rhs) of the rule. In TORPA format the arrow \( \rightarrow \) is written by the two symbols \( \Rightarrow \). A string \( s \in \Sigma^* \) rewrites to a string \( t \in \Sigma^* \) with respect to an SRS \( R \), written as \( s \rightarrow_R t \) if strings \( u, v \in \Sigma^* \) and a rule \( l \rightarrow r \in R \) exist such that \( s = ulv \) and \( t = urv \).

An SRS \( R \) is called terminating if no infinite sequence \( t_1, t_2, t_3, \ldots \) exists such that \( t_i \rightarrow_R t_{i+1} \) for all \( i = 1, 2, 3, \ldots \). Termination is also called strong normalization; therefore the property of \( R \) being terminating is written as \( \text{SN}(R) \).

An SRS \( R \) is called terminating relative to an SRS \( S \), written as \( \text{SN}(R/S) \), if no infinite sequence \( t_1, t_2, t_3, \ldots \) exists such that

- \( t_i \rightarrow R, S t_{i+1} \) for all \( i = 1, 2, 3, \ldots \), and
- \( t_i \rightarrow_R t_{i+1} \) for infinitely many values of \( i \).

Sometimes the notation \( R/S \) is used for the rewrite relation \( \rightarrow_S^* \rightarrow_R \); it is easy to see that \( \text{SN}(R/S) \) coincides with termination of this rewrite relation. By definition \( \text{SN}(R/S) \) and \( \text{SN}(R/(S \setminus R)) \) are equivalent. Therefore we will use the notation \( \text{SN}(R/S) \) only for \( R \) and \( S \) being disjoint. In writing an SRS \( R \cup S \) for which we want to prove \( \text{SN}(R/S) \) we write the rules of \( R \) by \( l \rightarrow r \) and the rules of \( S \) by \( l \rightarrow= r \). In TORPA format the arrow \( \rightarrow= \) is written by the three symbols \( \longrightarrow \). The rules from \( R \) are called strict rules; the rules from \( S \) are called non-strict rules.

First we prove a simple but very fruitful theorem for stepwise proving relative termination.

**Theorem 1** Let \( R, S, R' \) and \( S' \) be SRSs for which

- \( R \cup S = R' \cup S' \),
• $R \cap S = R' \cap S' = \emptyset$,
• $\text{SN}(R'/S')$, and
• $\text{SN}((R \cap S')/(S \cap S'))$.

Then $\text{SN}(R/S)$.

Proof: Assume any infinite reduction with respect to $R \cup S = R' \cup S'$. Since $\text{SN}(R'/S')$ after finitely many steps this reduction only consists of $S'$-steps. Since $\text{SN}((R \cap S')/(S \cap S'))$ after again finitely many steps the remaining part consists only of $S \cap S'$-steps, all being $S$-steps. Since $R \cap S = \emptyset$ we conclude that the assumed infinite reduction contains only finitely many $R$-steps. Hence we proved $\text{SN}(R/S)$. \hfill \Box

The way we will use Theorem 1 is as follows. If we have to prove $\text{SN}(R/S)$ then we try to split up $R \cup S$ into two disjoint parts $R'$ and $S'$ for which $R' \neq \emptyset$ and $\text{SN}(R'/S')$. If this succeeds then we may weaken the proof obligation $\text{SN}(R/S)$ to $\text{SN}((R \cap S')/(S \cap S'))$, i.e., all rules from $R'$ may be removed. This process is repeated as long as it is applicable. If after a number of steps $R \cap S' = \emptyset$ then $\text{SN}((R \cap S')/(S \cap S'))$ trivially holds and the desired proof has been given.

As an example we consider the SRS consisting of the following three rules:

$$ab \rightarrow ba, \ bc \rightarrow cb, \ ca \rightarrow ac.$$  

We want to prove termination of this SRS. Doing this in one step is not that easy, but proving $\text{SN}(R'/S')$ turns out to be very simple for $R'$ consisting of the last rule and $S'$ consisting of the first two rules, as we will see in the next section. Hence by Theorem 1 the last rule may be removed and it suffices to prove termination of the first two rules. In TORPA this will be done by another two applications of Theorem 1 each removing one rule. Then termination of the original SRS has been proved since no rules remain.

Next we give some basic observations on reversing strings. For a string $s$ write $s^{\text{rev}}$ for its reverse. For an SRS $R$ write

$$R^{\text{rev}} = \{ l^{\text{rev}} \rightarrow r^{\text{rev}} \mid l \rightarrow r \in R \}.$$  

Lemma 2 Let $R$ and $S$ be disjoint SRSs. Then $\text{SN}(R/S)$ if and only if $\text{SN}(R^{\text{rev}}/S^{\text{rev}})$.

Proof: This follows from the observation that if $s \rightarrow R t$ for any SRS $R$ then $s^{\text{rev}} \rightarrow R^{\text{rev}} t^{\text{rev}}$. \hfill \Box

Lemma 2 is strongly used in TORPA: if $\text{SN}(R/S)$ has to be proved then all techniques are not only applied on $R/S$ but also on $R^{\text{rev}}/S^{\text{rev}}$. Where 100 is the default basic number of attempts done in trying to find a suitable labelling in a fixed setting, the full number of attempts is 12 times this basic number or more before TORPA gives up. For each of these at least 1200 candidate SRSs both polynomial interpretations and recursive path order are tried, again both for the SRS itself and its reverse. Due to this number of candidate SRSs
and the fact that such a candidate SRS may have up to four times as many distinct symbols as the original SRS as we will see, in the implementation polynomial interpretations and recursive path order efficiency is important and extensive search should be avoided. As a rule of thumb we apply that $O(n^2m)$ is the highest allowed complexity where $n$ is the number of distinct symbols and $m$ is the size of the SRS.

Another way of reversing is given as follows: for any SRS $R$ define

$$R^{-1} = \{ r \rightarrow l \mid l \rightarrow r \in R \}.$$ 

It is not difficult to prove the following lemma.

**Lemma 3** Let $R, S$ be finite SRSs for which for every rule $l \rightarrow r \in R \cup S$ the sizes of $l$ and $r$ are equal. Then $\text{SN}(R/S)$ if and only if $\text{SN}(R^{-1}/S^{-1})$.

However, the requirement of all rules having both sides of equal length is that seldomly fulfilled that we decided not to use this lemma in TORPA.

## 3 Polynomial Interpretations

The ideas of (polynomial) interpretations go back to [12, 2]. First we give the underlying theory for doing this for string rewriting.

Let $A$ be a non-empty set and $\Sigma$ be an alphabet. Let $\epsilon$ denote the empty string in $\Sigma^*$. If $f_a : A \rightarrow A$ has been defined for every $a \in \Sigma$ then $f_s : A \rightarrow A$ is defined for every $s \in \Sigma^*$ inductively as follows:

$$f_\epsilon(x) = x$$ for every $x \in A$, 

$$f_{as}(x) = f_a(f_s(x))$$ for every $x \in A, a \in \Sigma, s \in \Sigma^*$.

**Theorem 4** Let $A$ be a non-empty set and let $>$ be a well-founded order on $A$. Let $f_a : A \rightarrow A$ be strictly monotone for every $a \in \Sigma$, i.e., $f_a(x) > f_a(y)$ for every $x, y \in A$ satisfying $x > y$. Let $R$ and $S$ be disjoint SRSs over $\Sigma$ such that

$$f_l(x) > f_r(x)$$

for all $x \in A$ and $l \rightarrow r \in R$, and

$$f_l(x) \geq f_r(x)$$

for all $x \in A$ and $l \rightarrow r \in S$. Then $\text{SN}(R/S)$.

**Proof:** Assume an infinite reduction $s_1 \rightarrow_{R \cup S} s_2 \rightarrow_{R \cup S} s_3 \rightarrow_{R \cup S} \cdots$. Let $i$ be any positive integer. Then $s_i = ulv$ and $s_{i+1} = urv$ for some $u, v \in \Sigma^*$ and $l \rightarrow r \in R \cup S$. Let $x \in A$ be arbitrary. Since all $f_a$ are strictly monotone, the function $f_u : A \rightarrow A$ is strictly monotone too. Hence if $l \rightarrow r \in R$ we obtain

$$f_{s_i}(x) = f_a(f_l(f_v(x))) > f_a(f_r(f_v(x))) = f_{s_{i+1}}(x),$$

where $s_i, s_{i+1} \in R \cup S$.

However, the requirement of all rules having both sides of equal length is that seldomly fulfilled that we decided not to use this lemma in TORPA.
and if $l \to r \in S$ we obtain
\[ f_{s_1}(x) = f_u(f_l(f_v(x))) \geq f_u(f_r(f_v(x))) = f_{s_{i+1}}(x). \]
Since $>$ is well-founded we conclude that the infinite inequality $f_{s_1}(x) \geq f_{s_2}(x) \geq f_{s_3}(x) \geq \cdots$ contains only finitely many strict inequalities, hence the infinite reduction $s_1 \to_{RJS} s_2 \to_{RJS} s_3 \to_{RJS} \cdots$ contains only finitely many $R$-steps, which we had to prove. \( \square \)

In the general case this approach is called *monotone algebras* ([14, 16]). In case $A$ consists of all integers $> N$ with the usual order for some number $N$, and the functions $f_a$ are polynomials this approach is called *polynomial interpretations*.

In TORPA only three distinct polynomials are used:

- the identity,
- the successor $\lambda x \cdot x + 1$, and
- $\lambda x \cdot 10x$.

For every symbol $a$ one of these three polynomials is chosen, and then it is checked whether using gives rise to $SN(R/S)$ for some non-empty $R$ according to Theorem 4. If so, then by using Theorem 1 the proof obligation can be weakened and the process is repeated. As a first example consider the output given by TORPA when entering the single rule $ab \to ba$:

TORPA 1.1 is applied to the string rewriting system
\[
ab \to ba
\]
Choose polynomial interpretation:
\[
a: \lambda x \cdot 10x, \text{ rest } \lambda x \cdot x + 1
\]
remove: $ab \to ba$
Terminating since no rules remain.

Here for $f_a$ and $f_b$ respectively $\lambda x \cdot 10x$ and the successor are chosen. Since
\[
f_{ab}(x) = f_a(f_b(x)) = 10(x + 1) > 10x + 1 = f_b(f_a(x)) = f_{ba}(x)
\]
for every $x$ indeed this single rule may be removed due to Theorem 4.

In forthcoming examples of TORPA output we will omit the first lines TORPA 1.1 is applied to $\cdots$.

As a similar example on relative termination rather than termination consider the two rules $ab \to ba, a \to= ca$, i.e., $SN(R/S)$ has to be proved where $R$ consists of the rule $ab \to ba$ and $S$ consists of the rule $a \to ca$. Now TORPA yields:

Choose polynomial interpretation:
\[
a: \lambda x \cdot 10x, \ b: \lambda x \cdot x + 1, \text{ rest identity}
\]
remove: $ab \to ba$
Relatively terminating since no strict rules remain.

As a next example consider the SRS mentioned in the previous section:
\[
ab \to ba, \ bc \to cb, \ ca \to ac.
\]
The reader is invited to apply his favorite technique or tool to this small SRS. As already indicated in the previous section TORPA repeats applying polynomial interpretations, yielding the following termination proof:

Choose polynomial interpretation:
\[c: \lambda x.10x, \ a: \lambda x.x+1, \ \text{rest identity}\]
remove: \(ca \rightarrow ac\)

Choose polynomial interpretation:
\[a: \lambda x.10x, \ b: \lambda x.x+1, \ \text{rest identity}\]
remove: \(ab \rightarrow ba\)

Choose polynomial interpretation:
\[b: \lambda x.10x, \ c: \lambda x.x+1, \ \text{rest identity}\]
remove: \(bc \rightarrow cb\)

Terminating since no rules remain.

Now we discuss the way how these polynomial interpretations are implemented in TORPA.

Checking whether \(f_l(x) > f_r(x)\) or \(f_l(x) \geq f_r(x)\) for all \(x\) for some rule \(l \rightarrow r\) is easily done: both \(f_l(x)\) and \(f_r(x)\) are of the shape \(mx+k\) for \(m > 0\) and \(k \geq 0\). We observe that \(mx+k > m'x+k'\) for all \(x \in A\) for a suitable \(A\) if and only if \(m > m' \vee (m = m' \wedge k > k')\). For instance, \(10x > x+19\) for all \(x > 3\), hence we may choose \(A\) to consist of all integers \(> 3\). In all cases the suitable set \(A\) will consist of all integers \(> N\); there is no need to compute \(N\) explicitly. The only thing to be done in checking whether \(f_l(x) > f_r(x)\) for all \(x\), for given \(f_a\) for every \(a \in \Sigma\), is computing \(m, k, m', k'\) such that \(f_l(x) = mx+k\) and \(f_r(x) = m'x+k'\), and check \(m > m' \vee (m = m' \wedge k > k')\). Checking whether \(f_l(x) \geq f_r(x)\) is done similarly.

However, doing this for all possible interpretations would be too expensive. Let \(n\) be the number of symbols. If for all symbols all three choices for polynomials would be tried there would be \(3^n\) attempts, being far beyond our desired efficiency bound \(O(n^2m)\). Therefore a selection among these \(3^n\) combinations had to be made. In order to reach the desired bound on the number of attempts all relevant choices are made for which at least \(n-2\) symbols have the same interpretations. These are:

- for respectively \(n, \ 1, \ 2, \ n-1\) or \(n-2\) symbols choose the successor, for the rest choose the identity;
- for one symbol choose \(\lambda x \cdot 10x\), from the remaining \(n-1\) symbols choose for respectively \(1, \ n-2\) or all symbols the successor, for the rest choose the identity;
- for two symbols choose \(\lambda x \cdot 10x\), for the rest choose the successor;
- for one or two symbols choose the successor, or for one the successor and for one the identity, and for the rest choose \(\lambda x \cdot 10x\).

For the cases in which \(\lambda x \cdot 10x\) is not involved essentially only a number of symbols is counted, and Lemma 2 is not applied since this would not increase the power. These cases
essentially correspond to Lemma 20 in [8]. For the other two cases Lemma 2 is applied: the interpretations are checked for both the SRS itself and its reverse.

Combining $\lambda x \cdot 10x$ with the identity without using successor does not make sense since it can easily be shown that if an interpretation of this type applies then also an interpretation applies combining only successor an identity.

To show how Lemma 2 is applied consider the single rule $ab \rightarrow baa$. Then TORPA yields:

Reverse every lhs and rhs of the system and choose polynomial interpretation:
- $b$: lambda $x.10x$
- $a$: lambda $x.x+1$
remove: $ab \rightarrow baa$
Terminating since no rules remain.

Indeed, without reversing no polynomial interpretation applies. In [14] it was proved that even no interpretation in the natural numbers applies. But after reversing we obtain

$$f_{ba}(x) = f_b(f_a(x)) = 10x + 10 > 10x + 2 = f_a(f_b(f_a(x))) = f_{aab}(x)$$

for all $x$.

The choice of the number 10 in $\lambda x \cdot 10x$ is quite arbitrary. One can make the artificial rule $ab \rightarrow bbbbbbbbbba$ which cannot be proved to be terminating by the present version of polynomial interpretations, and can be proved if $\lambda x \cdot 10x$ was replaced by $\lambda x \cdot N x$ for any $N > 10$. However, this rule with 10 consecutive $b$-s is that artificial that we do not worry about this.

As a last example in this section we consider the three rules

$$a \rightarrow bf, \ b \rightarrow c, \ cd \rightarrow dda.$$ 

Here TORPA yields:

Choose polynomial interpretation:
- $f$: identity, $d$: lambda $x.x+1$, rest lambda $x.10x$
remove: $cd \rightarrow dda$
Choose polynomial interpretation
- $a$: lambda $x.x+1$, rest identity
remove: $a \rightarrow bf$
Choose polynomial interpretation
- $b$: lambda $x.x+1$, rest identity
remove: $b \rightarrow c$
Terminating since no rules remain.

4 Recursive Path Order

Recursive path order is already an old technique too; it was introduced by Derschowitz [5]. Restricted to string rewriting it means that for a fixed order $>$ on the finite alphabet $\Sigma$, called the precedence, there is an order $>_rpo$ on $\Sigma^*$ called recursive path order. The main
property of this order that if \( l >_{rpo} r \) for all rules \( l \rightarrow r \) of an SRS \( R \), then \( R \) is terminating. This order \( >_{rpo} \) has the following defining property: \( s >_{rpo} t \) if and only if \( s \) can be written as \( s = as' \) for \( a \in \Sigma \), and either

- \( s' = t \) or \( s' >_{rpo} t \), or

- \( t \) can be written as \( t = bt' \) for \( b \in \Sigma \), and either
  - \( a > b \) and \( s >_{rpo} t' \), or
  - \( a = b \) and \( s' >_{rpo} t' \).

For further details we refer to [16]. The basic idea of proving termination of an SRS using recursive path order is that one starts with no restrictions on the order \( > \), and then for every rule \( l \rightarrow r \) one collects restrictions on \( > \) by which one can conclude \( l >_{rpo} r \) by the above defining property. If this succeeds for all rules without conflicting restrictions on \( > \), then termination has been proved. Here restrictions on \( > \) are conflicting if they violate transitivity and irreflexivity of \( > \). For instance, by the only restriction \( a > b \) which is not conflicting one easily derives \( ab >_{rpo} bba \) from the above defining property, hence proving \( \text{SN}\{ab \rightarrow bba\} \).

However, search for non-conflicting restrictions may include a lot of branching, causing this process being exponential. For a single term rewrite rule \( l \rightarrow r \) it has been proved in [10] that checking whether a precedence \( > \) exists satisfying \( l >_{rpo} r \) is NP-complete. Here we restrict to string rewriting, but encounter similar problems on efficiency. Therefore in TORPA instead of full recursive path order the alternative relation \( >'_{rpo} \) is used, defined by \( s >'_{rpo} t \) if and only if \( s \) can be written as \( s = as' \) for \( a \in \Sigma \), and either

- \( s' \geq t \), or

- \( t \) can be written as \( t = bt' \) for \( b \in \Sigma \), and either
  - \( a > b \) and \( s >'_{rpo} t' \), or
  - \( a = b \) and \( s' >'_{rpo} t' \).

Here \( s \geq t \) means that the string \( t \) can be obtained from \( s \) by removing zero or more symbols. It is easy to see that \( s \geq t \) implies \( s = t \) or \( s >_{rpo} t \), hence \( >'_{rpo} \subseteq >_{rpo} \).

If after application of polynomial interpretations \( \text{SN}(R/S) \) has to be proven then TORPA tries to find a precedence \( > \) such that \( l >'_{rpo} r \) for all rules \( l \rightarrow r \in R \), and \( l >'_{rpo} r \) or \( l = r \) for all rules \( l \rightarrow r \in S \). For every rule this yields a number of restrictions of the shape \( a > b \). Finally it is checked whether these restrictions violate transitivity and reflexivity of \( > \), i.e., it is checked whether the corresponding directed graph contains a cycle. If not, then \( \text{SN}(R/S) \) has been proved. All of these steps can be done efficiently.

For instance, on the single rewrite rule \( abc \rightarrow bacb \) TORPA yields:

Terminating by recursive path order with precedence:

\[
\begin{align*}
a &> b \\
b &> c
\end{align*}
\]
Our version $\succ_{rpo}$ is weaker than $\succ_{rpo}$. For instance, for $b > c > a$ we have $abc >_{rpo} cbba$ while no precedence $>$ exists satisfying $abc >_{rpo} cbba$. However, in TORPA such a rule $abc \rightarrow cbba$ is no problem since then polynomial interpretations apply. In fact we could not find any example for which full recursive path order would succeed while TORPA fails. Therefore the present efficient version $\succ_{rpo}$ is kept as the basic version of recursive path order which is applied a number of times as follows.

For a symbol $a$ and an SRS $R$ write $\text{rema}(R)$ for the SRS obtained by removing all occurrences of $a$ from all left hand sides and right hand sides of rules in $R$. It is easy to see that SN($R/S)$ can be concluded from SN($\text{rema}(R)/\text{rema}(S)$). If there are $n$ symbols the version of recursive path order as described above is applied $2n + 2$ times in TORPA as long as no proof is found: for $R/S$ and $R^{rev}/S^{rev}$, and for $\text{rema}(R)/\text{rema}(S)$ and $\text{rema}(R^{rev})/\text{rema}(S^{rev})$ for all symbols $a$. For instance, for the SRS consisting of the two rules

$$apbc \rightarrow caqdbapbap, \quad poqd \rightarrow daqdbapbap$$

TORPA yields:

**Terminating since rev(1) > rev(r)** for all rules $1 \rightarrow r$

where $>$ is the recursive path order with precedence:

c>p c>a c>b q>p q>a q>b, after removing $d$

Experience shows that most termination proofs generated by TORPA do not use recursive path order. One reason for this is that first always polynomial interpretations are tried, and polynomial interpretations are often successful for SRSs for which recursive path order would be successful too. Several extensions of recursive path order are possible, for instance using a precedence in which symbols can be equivalent. However, for the full general version of this extension we again would have efficiency problems; remember that typically 1200 attempts are done on SRSs having up to $4n$ distinct symbols if the original SRS has $n$ distinct symbols. Moreover, many instances of this extension are already covered by polynomial interpretations. For instance, if the alphabet splits up in two equivalence classes $C_1 > C_2$ then up to exceptional rules like $ab \rightarrow bbbbbbbba$ also a polynomial interpretation can be found by choosing $f_a = \lambda x \cdot 10x$ for $a \in C_1$ and $f_a = \lambda x \cdot x + 1$ for $a \in C_2$.

## 5 Semantic Labelling

The technique of semantic labelling was introduced in [15]. Here we restrict to the version for string rewriting in which every symbol is labelled by the value of its argument. For this version we will present the theory for relative termination. Then we will show how this theory is applied in TORPA for the case of (quasi-)models containing only two elements.

Fix a non-empty set $A$ and maps $f_a : A \rightarrow A$ for all $a \in \Sigma$ for some alphabet $\Sigma$. Let $f_s$ for $s \in \Sigma^*$ be defined as before. Let $\overline{\Sigma}$ be the alphabet consisting of the symbols $a_x$ for $a \in \Sigma$ and $x \in A$. The labelling function $\text{lab} : \Sigma^* \times A \rightarrow \overline{\Sigma}$ is defined inductively as follows:

$$\text{lab}(\epsilon, x) = \epsilon \quad \text{for } x \in A,$$
\[
\text{lab}(sa, x) = \text{lab}(s, f_a(x))a_x \quad \text{for } s \in \Sigma^*, a \in \Sigma, x \in A.
\]

For an SRS \( R \) over \( \Sigma \) define
\[
\text{lab}(R) = \{ \text{lab}(l, x) \rightarrow \text{lab}(r, x) \mid l \rightarrow r \in R, x \in A \}.
\]

**Theorem 5** Let \( R \) and \( S \) be two disjoint SRSs over an alphabet \( \Sigma \). Let \( > \) be a well-founded order on a non-empty set \( A \). Let \( f_a : A \rightarrow A \) be defined for all \( a \in \Sigma \) such that

- \( f_a(x) \geq f_a(y) \) for all \( a \in \Sigma, x, y \in A \) satisfying \( x > y \), and
- \( f_l(x) \geq f_r(x) \) for all \( l \rightarrow r \in R \cup S, x \in A \).

Let Dec be the SRS over \( \overline{\Sigma} \) consisting of the rules \( a_x \rightarrow a_y \) for all \( a \in \Sigma, x, y \in A \) satisfying \( x > y \).

Then \( \text{SN}(R/S) \) if and only if \( \text{SN}(\text{lab}(R)/(\text{lab}(S) \cup \text{Dec})) \).

**Proof:** For the only-if-part assume \( \text{SN}(R/S) \) and the existence of an infinite reduction of \( \text{lab}(R) \cup \text{lab}(S) \cup \text{Dec} \) containing infinitely many \( \text{lab}(R) \) steps. Remove all subscripts in this reduction, then every \( \text{lab}(R) \) step transforms to an \( R \)-step, every \( \text{lab}(S) \) step transforms to an \( S \)-step, and every Dec-step transforms to equality, contradicting \( \text{SN}(R/S) \).

For the converse we consider the following claim.

**Claim:** If \( x \in A \) and \( s \rightarrow_R t \) then \( \text{lab}(s, x) \rightarrow_{\text{lab}(R)} \rightarrow_{\text{Dec}} \text{lab}(t, x) \).

Choose \( x \in A \) arbitrary. According to this claim applying \( \text{lab}(\cdot, x) \) to all strings in an infinite \( R \cup S \) reduction containing infinitely many \( R \) steps gives rise to an infinite \( \text{lab}(R) \cup \text{lab}(S) \cup \text{Dec} \) reduction containing infinitely many \( \text{lab}(R) \) steps, concluding the proof of the theorem. Hence it remains to prove the claim. In order to do so we first give two basic properties that we need:

- if \( u, v \in \Sigma^* \) and \( x \in A \) then \( \text{lab}(uv, x) = \text{lab}(u, f_v(x))\text{lab}(v, x) \), and
- if \( u \in \Sigma^* \) and \( x, y \in A, x \geq y \), then \( \text{lab}(u, x) \rightarrow_{\text{Dec}} \text{lab}(u, y) \).

Both properties can be checked directly; for the last one the condition \( f_a(x) \geq f_a(y) \) for \( x > y \) is essential.

For \( s \rightarrow_R t \) we can write \( s = ulv \) and \( t = urv \) for \( u, v \in \Sigma^* \) and \( l \rightarrow r \in R \). Using the above basic properties and the conditions of the theorem we obtain

\[
\text{lab}(s, x) = \text{lab}(ulv, x)
= \text{lab}(u, f_v(x))\text{lab}(l, f_v(x))\text{lab}(v, x)
\rightarrow_{\text{lab}(R)} \text{lab}(u, f_v(x))\text{lab}(r, f_v(x))\text{lab}(v, x)
= \text{lab}(u, f_v(x))\text{lab}(r, f_v(x))\text{lab}(v, x)
\rightarrow_{\text{Dec}} \text{lab}(u, f_v(x))\text{lab}(r, f_v(x))\text{lab}(v, x)
= \text{lab}(uv, x)
= \text{lab}(t, x),
\]
concluding the proof.

In case the relation $>$ is empty the set $A$ together with the functions $f_a$ for $a \in \Sigma$ is called a model for the SRS, otherwise it is called a quasi-model. It is called a model since then for every rule $l \rightarrow r$ the interpretation $f_l$ of $l$ is equal to the interpretation $f_r$ of $r$. Note that $\text{Dec} = \emptyset$ in case of a model.

The main application of Theorem 5 in TORPA is that if $\text{SN}(R/S)$ has to be proven then for a number of (quasi-)model candidates it is tried to prove $\text{SN}(\text{lab}(R)/(\text{lab}(S) \cup \text{Dec}))$. If this succeeds we are done. For instance, if we enter the single rule $aa \rightarrow aba$ then TORPA may yield:

Apply labelling with the following interpretation in $\{0,1\}$:

- $a$: constant 0
- $b$: constant 1

and label every symbol by the value of its argument.

This interpretation is a model.

Labelled system:

- $a0 \ a0 \rightarrow a1 \ b0 \ a0$
- $a0 \ a1 \rightarrow a1 \ b0 \ a1$

Choose polynomial interpretation

- $a0 : \lambda x.x+1$, rest identity
- remove: $a0 \ a0 \rightarrow a1 \ b0 \ a0$
- remove: $a0 \ a1 \rightarrow a1 \ b0 \ a1$

Terminating since no rules remain.

In the notation of Theorem 5 this means that $A = \{0,1\}$, $f_a(x) = 0$ and $f_b(x) = 1$ for $x \in A$, $R = \{aa \rightarrow aba\}$, $S = \text{lab}(S) = \text{Dec} = \emptyset$. Since $\text{lab}(aa, x) = a_0a_x$ and $\text{lab}(aba, x) = a_1b_0a_x$ for $x = 0,1$, the labelled system $\text{lab}(R)$ is as indicated, for which indeed a termination proof is found.

In TORPA Theorem 5 is only applied for the case where $A$ consists of two elements, named 0 and 1. For every symbol $a$ there are four possibilities for $f_a : A \rightarrow A$:

$$f_a = \lambda x \cdot x, \quad f_a = \lambda x \cdot 0, \quad f_a = \lambda x \cdot 1, \quad f_a = \lambda x \cdot 1 - x.$$ 

Up to renaming this set $A = \{0,1\}$ admits only two strict orders $> : > = \emptyset$ and $> = (1,0)$. For the first one (the model case) for all symbols $a$ all four interpretations for $f_a$ are allowed, and the only restriction is that $f_l = f_r$ for all rules $l \rightarrow r \in R \cup S$. For the second order (the quasi-model case) for all symbols $a$ only the first three interpretations for $f_a$ are allowed, since $f_a = \lambda x \cdot 1 - x$ does not satisfy the requirement that $f_a(x) \geq f_a(y)$ for $x > y$. On the other hand, now the restriction on the rules is weaker: it should hold that $f_l(x) \geq f_r(x)$ for all rules $l \rightarrow r \in R \cup S$ and $x \in A$.

Disallowing $f_a = \lambda x \cdot 1 - x$ in the quasi-model case is essential for validity of Theorem 5. For instance, consider the SRS $R$ consisting of the two rules $aba \rightarrow abba$ and $ba \rightarrow a$, and $S = \emptyset$. Clearly $R$ is not terminating. By choosing $f_a = \lambda x \cdot 0$ and $f_b = \lambda x \cdot 1 - x$ we
indeed have $f_l(x) \geq f_r(x)$ for both rules $l \rightarrow r$ and $x \in A$. However, $SN(lab(R)/Dec)$ is easily proved by TORPA.

Now TORPA works as follows. First the model approach is tried. For SRSs $R, S$ for which $SN(R/S)$ has to be proved, for all $a \in \Sigma$ the functions $f_a$ are chosen randomly among the four possible functions until $f_l = f_r$ for all rules $l \rightarrow r \in R \cup S$. Since such a model always exists (for instance, $f_a = \lambda x \cdot x$ for all $a \in \Sigma$ always yields a model), such a model will always be found. Then for this choice $lab(R)$ and $lab(S)$ are computed, and it is tried to prove $SN(lab(R)/lab(S))$ by means of polynomial interpretations and recursive path order. If this succeeds the desired proof is generated, otherwise the whole procedure is repeated. There is a basic maximal number of attempts to be done. The default of this number is 100.

It can be the case that there are different solutions. For instance, for the single rule $aa \rightarrow aba$ we saw that choosing constant 0 for $f_a$ and constant 1 for $f_b$ was successful. However, $f_a = \lambda x \cdot 1$ and $f_b = \lambda x \cdot 1 - x$ is successful too, just like a few other variants. Subsequent attempts to prove termination by TORPA may yield different solutions, due to the use of the random generator.

In case this first series of attempts was not yet successful a similar procedure is applied for quasi-models: now for all $a \in \Sigma$ the functions $f_a$ are chosen randomly among the three allowed functions until $f_l(x) \geq f_r(x)$ for all rules $l \rightarrow r \in R \cup S$ and $x \in A$. Again such a quasi-model always exists and hence will always be found. Then it is tried to prove $SN(lab(R)/(lab(S) \cup Dec))$ by means of polynomial interpretations and recursive path order. Again this is repeated up to a number of times corresponding to the basic maximal number.

For both the model case and the quasi-model case everything is done twice as long as no solution is found: once for $R/S$ and once for $R^{rev}/S^{rev}$. As an example, we consider the SRS consisting of the four rules

$$a \rightarrow bc, \ ab \rightarrow ba, \ dc \rightarrow da, \ ac \rightarrow ca.$$ 

Now the output of TORPA starts by
Reverse every lhs and rhs of the system.

Apply labelling with the following interpretation in \{0,1\}:
- a: identity
- b: constant 0
- c: identity
- d: constant 1

and label every symbol by the value of its argument.

This is a quasi-model for 1 > 0.

Labelled system:

\[
\begin{align*}
& a_0 \rightarrow c_0 \ b_0 \\
& a_1 \rightarrow c_0 \ b_1 \\
& b_0 \ a_0 \rightarrow a_0 \ b_0 \\
& b_1 \ a_1 \rightarrow a_0 \ b_1 \\
& c_1 \ d_0 \rightarrow a_1 \ d_0 \\
& c_1 \ d_1 \rightarrow a_1 \ d_1 \\
& c_0 \ a_0 \rightarrow a_0 \ c_0 \\
& c_1 \ a_1 \rightarrow a_1 \ c_1 \\
& a_1 \rightarrow a_0 \\
& b_1 \rightarrow b_0 \\
& c_1 \rightarrow c_0 \\
& d_1 \rightarrow d_0
\end{align*}
\]

and ends by a standard termination proof by polynomial interpretations of this labelled system. We are not aware of any other technique by which termination of this SRS can be proved.

Still this is not yet the whole story. It can be the case that some attempt to prove SN(R/S) by means of proving SN(lab(R)/(lab(S) \cup Dec)) fails, but applying polynomial interpretations succeeds in removing some rules. If the remaining rules are all contained in lab(R') \cup lab(S') \cup Dec for R' \subseteq R and S' \subseteq S, at least one of the inclusions being strict, then by applying Theorem 5 in the reverse direction it suffices to prove SN(R'/S'). In fact this is removal of labels. In the default setting of TORPA in trying to apply labelling it is checked every time whether this removal of labels can be applied. If so, then all labelling is forgotten and TORPA starts from scratch trying to prove SN(R'/S'). For instance, applying TORPA in the default setting to the two rules

\[
aba \rightarrow abba, \ bab \rightarrow baab
\]

yields
Apply labelling with the following interpretation in \(\{0,1\}\):

\[
\begin{align*}
a & : \lambda x.1-x \\
b & : \text{constant } 0
\end{align*}
\]

and label every symbol by the value of its argument.

This interpretation is a model.

Labelled system:

\[
\begin{align*}
a0 \; b1 \; a0 & \rightarrow a0 \; b0 \; b1 \; a0 \\
a0 \; b0 \; a1 & \rightarrow a0 \; b0 \; b0 \; a1 \\
b1 \; a0 \; b0 & \rightarrow b0 \; a1 \; a0 \; b0 \\
b1 \; a0 \; b1 & \rightarrow b0 \; a1 \; a0 \; b1
\end{align*}
\]

Choose polynomial interpretation

\[
b1 : \lambda x.x+1, \text{ rest identity}
\]

remove: \(b1 \; a0 \; b0 \rightarrow b0 \; a1 \; a0 \; b0\)

remove: \(b1 \; a0 \; b1 \rightarrow b0 \; a1 \; a0 \; b1\)

Remaining rules:

\[
\begin{align*}
a0 \; b1 \; a0 & \rightarrow a0 \; b0 \; b1 \; a0 \\
a0 \; b0 \; a1 & \rightarrow a0 \; b0 \; b0 \; a1
\end{align*}
\]

Remove all labels, remaining unlabelled system:

\(aba \rightarrow abba\)

Apply labelling with the following interpretation in \(\{0,1\}\):

\[
\begin{align*}
a & : \text{constant } 0 \\
b & : \lambda x.1-x
\end{align*}
\]

and label every symbol by the value of its argument.

This interpretation is a model.

Labelled system:

\[
\begin{align*}
a1 \; b0 \; a0 & \rightarrow a0 \; b1 \; b0 \; a0 \\
a1 \; b0 \; a1 & \rightarrow a0 \; b1 \; b0 \; a1
\end{align*}
\]

Choose polynomial interpretation

\[
a1 : \lambda x.x+1, \text{ rest identity}
\]

remove: \(a1 \; b0 \; a0 \rightarrow a0 \; b1 \; b0 \; a0\)

remove: \(a1 \; b0 \; a1 \rightarrow a0 \; b1 \; b0 \; a1\)

Terminating since no rules remain.

Without removal of labels a termination proof could not have been given using the techniques described until now. In TORPA you may choose whether this removal of labels should be tried or not. In the version with the graphical user interface this is done by a check box that will be visible when pushing the button ‘customize’; in the plain version this is done by adjusting the input format according the instructions given on www.win.tue.nl/~hzantema/torpa.html.

If a termination proof can be found without the option of removal of labels then a termination proof can be found too with the option. This is due to the fact that both polynomial interpretations and recursive path order are robust under removal of rules.
Therefore the option of removal of labels is the default. However, there are examples by which the full termination proof is longer and therefore harder to verify if the option is on. For instance, this holds for the SRS

\[
\begin{align*}
t f & \rightarrow t c n \\
n f & \rightarrow f n \\
o f & \rightarrow f o \\
n s & \rightarrow f s \\
o s & \rightarrow f s \\
c f & \rightarrow f c \\
c n & \rightarrow n c \\
c o & \rightarrow o c \\
c o & \rightarrow o.
\end{align*}
\]

This is essentially the rewrite system as it appears in the first case study of liveness in [8]: termination of this rewrite system implies that all old processes in a waiting line will eventually be served. Surprisingly, by putting the option off for removal of labels, TORPA yields exactly the same labelling proof as was given in [8] guarded by some heuristics.

6 Dependency Pairs

The technique of dependency pairs was introduced in [1] and is extremely useful for automatically proving termination of term rewriting. Here we only use a mild version of it, without explicitly doing argument filtering or dependency graph approximation. It turns out that often the same reduction of the problem caused by these more involved parts of the dependency pair technique is done by applying our versions of labelling and polynomial interpretations. Since we want to present dependency pairs in the terminology of relative termination being quite different from the original presentation in [8], and we want to be self-contained, we present all required theory here. In our approach we do not even need the basic notion of chains of dependency pairs.

Fix $R$ to be an SRS over an alphabet $\Sigma$. Let $\Sigma_D$ be the set of defined symbols of $R$, i.e., the set of symbols occurring as the leftmost symbol of the left hand side of a rule in $R$. For every defined symbol $a \in \Sigma_D$ we introduce a fresh symbol $\bar{a}$. In case $a$ is a lowercase symbol then usually its capital version is used as the notation for $\bar{a}$. In TORPA this convention is followed; if $a$ is not a lowercase symbol then instead the symbol # is put in front.

Write $\bar{\Sigma} = \Sigma \cup \{\bar{a} \mid a \in \Sigma_D\}$. The SRS $DP(R)$ over $\bar{\Sigma}$ is defined to consist of all rules of the shape

\[\bar{a}l' \rightarrow \bar{b}r''\]

for which $al' = l$ and $r = r'br''$ for some rule $l \rightarrow r$ in $R$ and $a, b \in \Sigma_D$. Rules of $DP(R)$ are called dependency pairs. In our view the main theorem of dependency pairs states that $SN(R)$ if and only if $SN(DP(R)/R)$. It is used in TORPA as follows: if proving $SN(R)$ does not succeed by the earlier techniques, then these techniques are applied to trying to prove $SN(DP(R)/R)$. If this again does not succeed then finally they are applied to trying to prove $SN(DP(R^{rev})/R^{rev})$. In fact the desire for being able to do so was one of the main reasons to generalize the basic methods to relative termination and design TORPA to cover relative termination.

As a first example on dependency pairs consider the two rules $ab \rightarrow c, c \rightarrow ba$. Here TORPA yields
Dependency pair transformation:
ab \rightarrow c
\text{Choose polynomial interpretation}
b \text{c: lambda } x.x+1, \text{ rest identity}
remove: Ab \rightarrow C
\text{Choose polynomial interpretation}
C \text{ lambda } x.x+1, \text{ rest identity}
remove: C \rightarrow A
Relatively terminating since no strict rules remain.

In order to prove the main theorem we first need a lemma. For a string \( s \) we write \( \infty(s) \) if there is an infinite \( R \)-reduction starting in \( s \), and \( \text{SN}(s) \) if not.

**Lemma 6** Let \( s \in \Sigma^* \) satisfy \( \infty(s) \). Then \( s \) can be written as \( s = uav \) for \( a \in \Sigma_D \), \( u, v \in \Sigma^* \) such that \( w, x \in \Sigma^* \) and \( ltor \in R \) exist such that \( \text{SN}(v), v \rightarrow_R w, aw = lx \) and \( \infty(rx) \).

**Proof:** Choose \( s = uav \) for \( a \in \Sigma, u, v \in \Sigma^* \) such that \( \text{SN}(v) \) and \( \infty(av) \). Then an infinite reduction of \( av \) should contain a reduction step on its leftmost position. Let \( lx \rightarrow_R rx \) be the first one. Then all desired properties hold.

**Theorem 7** Let \( R \) be any SRS. Then \( \text{SN}(R) \) if and only if \( \text{SN}(DP(R)/R) \).

**Proof:** For the if-part assume \( \infty(s) \) for \( s \in \Sigma^* \); we have to prove that there is an infinite \( DP(R) \cup R \)-reduction containing infinitely many \( DP(R) \) steps. Applying Lemma 6 to \( s \) yields \( s = uav \) for which \( \infty(av) \) and \( \text{SN}(v) \). We will show that \( av \) starts an infinite \( DP(R) \cup R \)-reduction containing infinitely many \( DP(R) \) steps. This follows from the following claim.

**Claim:** If \( \infty(av) \) and \( \text{SN}(v) \) then \( a', v' \) exist satisfying \( \infty(a'v') \), \( \text{SN}(v') \) and
\[
\overline{av} \rightarrow^*_{DP(R)} \overline{a'v'}.
\]

We prove this claim by applying Lemma 6 on \( av \). Since \( \text{SN}(v) \) the resulting \( u \) is empty, and we obtain \( w, x, l \rightarrow_r \) such that \( v \rightarrow_R w, aw = lx \) and \( \infty(rx) \). Write \( l = al_0 \). Next we apply Lemma 6 on \( rx \), yielding \( rx = u'a'v', \text{SN}(v'), v' \rightarrow^*_R w', a'w' = l'x', \) and \( \infty(r'x') \). By \( a'v' \rightarrow^*_R a'w' = l'x' \rightarrow_R r'x' \) we conclude \( \infty(a'v') \). If \( \#a' \geq \#r \) then \( x \) can be written as \( x = u''a'v' \) implying an infinite reduction \( v \rightarrow^*_R w = l_0x = l_0u''a'v' \rightarrow_R \cdots \), contradicting \( \text{SN}(v) \). Hence \( \#a' < \#r \), by which we can write \( r' = u'a'r_0 \). Since \( a' \) is the leftmost symbol of \( l' \) we have \( a' \in \Sigma_D \). Hence \( al_0 \rightarrow a'l_0 \) is a rule from \( DP(R) \). We conclude
\[
\overline{av} \rightarrow^*_R \overline{aw} = \overline{al_0x} \rightarrow_{DP(R)} \overline{a'l_0x} = \overline{a'v'},
\]
concluding the proofs of the claim and the if-part of the theorem.
For the only-if-part we prove assume \( \text{SN}(R) \) and prove \( \text{SN}(DP(R) \cup R) \), from which \( \text{SN}(DP(R)/R) \) immediately follows. Due to type elimination / type introduction ([14]) \( \text{SN}(DP(R) \cup R) \) is equivalent to termination of well-typed reductions, for the typing \( a : s \rightarrow s \) for \( a \in \Sigma \) and \( \bar{a} : s \rightarrow t \) for \( a \in \Sigma_D \). Reductions of type \( s \) are \( R \)-reductions that terminate by assumption. Assume an infinite reduction of type \( t \) exists, then every string in this reduction is of the shape \( \bar{a}u \) for \( a \in \Sigma_D, u \in \Sigma^* \). By removing over-lines and adding the string \( r' \) from the rule \( \bar{a}' \rightarrow r'br'' \) when the rule \( \bar{a}r \rightarrow br' \) from \( DP(R) \) is applied, this infinite \( DP(R) \)-reduction transforms to an infinite \( R \)-reduction, contradicting \( \text{SN}(R) \). \( \Box \)

Note that for proving termination by dependency pairs only the if-part of Theorem 7 is used; the only-if-part is only included for completeness of the theory.

The standard dependency pair approach as proposed in [1] essentially coincides with the analysis of infinite well-typed reductions of type \( t \), called \emph{chains}.

It is a natural question whether dependency pairs can be used for relative termination rather than termination. More precisely, if we want to prove \( \text{SN}(R/S) \) and all other methods fail then we would try to prove \( \text{SN}(T_1(R, S)/T_2(R, S)) \) for transformations \( T_1, T_2 \) for which we have the theorem

\[
\text{SN}(T_1(R, S)/T_2(R, S)) \Rightarrow \text{SN}(R/S).
\]

A first guess would be \( T_1(R, S) = DP(R) \) and \( T_2(R, S) = R \cup S \cup DP(S) \), where \( DP \) is defined with respect to the defined symbols of \( R \cup S \). However, then the desired theorem does not hold. For instance, let \( R \) consist of the rule \( a \rightarrow b \) and \( S \) of the rule \( f(b) \rightarrow f(a) \). Then clearly \( \text{SN}(R/S) \) does not hold, while \( \text{SN}(DP(R)/\cdots) \) holds by the trivial reason that \( DP(R) = \emptyset \).

Instead a valid instance of the desired theorem is

\[
\text{SN}(DP(R \cup S)/(R \cup S)) \Rightarrow \text{SN}(R/S).
\]

This property immediately follows from Theorem 7 and the trivial implication \( \text{SN}(R \cup S) \Rightarrow \text{SN}(R/S) \). This is applied in TORPA: if \( \text{SN}(R/S) \) has to be proved and all other techniques fail then it is tried to prove \( \text{SN}(DP(R \cup S)/(R \cup S)) \). However, this only may succeed if \( \text{SN}(R \cup S) \) holds, which is usually not the case. We leave as an open problem to find a non-trivial variant of the dependency pair transformation able to prove \( \text{SN}(R/S) \) in case \( \text{SN}(R \cup S) \) does not hold.

TORPA may find termination proofs of the following shape:

- First zero or more rules are removed by polynomial interpretations.
- Then the dependency pair transformation is applied on the SRS or its reverse.
- Then a labelling is applied followed by repeated polynomial interpretations. After removing labels this step may be repeated a number of times.
- Finally the termination proof of the original SRS is completed if after repeated polynomial interpretations no strict rules remain.
In termination proofs found by TORPA combining labelling and dependency pairs always the dependency pair transformation is applied first. It is also valid to do it the other way around: first apply labelling and then do the dependency pair transformation. The implementation of TORPA was extended in such a way that the dependency pair transformation was applied on the labelled systems. However, we did not find any example for which a termination proof was found using this extension while it was not found by the version without this extension. Therefore this extension has been removed from the present version.

We conclude this section by an example where TORPA finds a termination proof by combining most of the techniques described in this paper. Due to the use of a random generator in search for a labelling it may occur that on different platforms or in subsequent attempts slightly different proofs are found. We apply TORPA on the SRS consisting of the four rules

\[ bca \rightarrow ababc, \ b \rightarrow cc, \ cd \rightarrow abca, \ aa \rightarrow acba. \]

First the polynomial interpretation \( d : \lambda x \cdot x + 1 \), rest identity, is chosen, by which the third rule is removed. Then the remaining SRS is reversed and the dependency pair transformation is applied, yielding the 10 rules

\[
\begin{align*}
acb & \rightarrow cbaba \\
b & \rightarrow cc \\
aa & \rightarrow abca \\
Acb & \rightarrow Baba \\
Acb & \rightarrow Aba \\
Acb & \rightarrow Ba \\
Acb & \rightarrow A \\
Aa & \rightarrow Abca \\
Aa & \rightarrow Bca \\
Aa & \rightarrow A
\end{align*}
\]

By polynomial interpretations three rules are removed, and on the remaining 7 rules a quasi-model labelling is found, giving a labelled SRS consisting of 20 rules:

\[
\begin{align*}
\text{Choose polynomial interpretation} & \quad A: \lambda x \cdot x + 1, \text{ rest identity} \\
\text{remove:} & \quad Acb \rightarrow Baba \\
\text{remove:} & \quad Acb \rightarrow Ba \\
\text{remove:} & \quad Aa \rightarrow Bca \\
\text{Remaining rules:} & \\
acb & \rightarrow cbaba \\
b & \rightarrow cc \\
aa & \rightarrow abca \\
Acb & \rightarrow Aba \\
Acb & \rightarrow A \\
Aa & \rightarrow Abca \\
Aa & \rightarrow A
\end{align*}
\]
Apply labelling with the following interpretation in \{0,1\}:

- **b**: constant 1
- **c**: constant 0
- **a**: constant 0
- **d**: constant 0
- **B**: constant 1
- **A**: constant 0

and label every symbol by the value of its argument.

This is a quasi-model for \(1 > 0\).

Labelled system:

\[
\begin{align*}
\text{a0} &\text{ c1 b0} \implies \text{ c1 b0 a1 b0 a0} \\
\text{a0} &\text{ c1 b1} \implies \text{ c1 b0 a1 b0 a1} \\
\text{b0} &\implies \text{ c0 c0} \\
\text{b1} &\implies \text{ c0 c1} \\
\text{a0} &\text{ a0} \implies \text{ a1 b0 c0 a0} \\
\text{a0} &\text{ a1} \implies \text{ a1 b0 c0 a1} \\
\text{A0} &\text{ c1 b0} \implies \text{ A1 b0 a0} \\
\text{A0} &\text{ c1 b1} \implies \text{ A1 b0 a1} \\
\text{A0} &\text{ c1 b0} \implies \text{ A0} \\
\text{A0} &\text{ c1 b1} \implies \text{ A1} \\
\text{A0} &\text{ a0} \implies \text{ A1 b0 c0 a0} \\
\text{A0} &\text{ a1} \implies \text{ A1 b0 c0 a1} \\
\text{A0} &\text{ a0} \implies \text{ A0} \\
\text{A0} &\text{ a1} \implies \text{ A1} \\
\text{b1} &\implies \text{ b0} \\
\text{c1} &\implies \text{ c0} \\
\text{a1} &\implies \text{ a0} \\
\text{d1} &\implies \text{ d0} \\
\text{B1} &\implies \text{ B0} \\
\text{A1} &\implies \text{ A0}
\end{align*}
\]

By polynomial interpretations 10 of them are removed, yielding

\[
\begin{align*}
\text{a0} &\text{ c1 b0} \implies \text{ c1 b0 a1 b0 a0} \\
\text{b0} &\implies \text{ c0 c0} \\
\text{a0} &\text{ a0} \implies \text{ a1 b0 c0 a0} \\
\text{a0} &\text{ a1} \implies \text{ a1 b0 c0 a1} \\
\text{A0} &\text{ a0} \implies \text{ A1 b0 c0 a0} \\
\text{A0} &\text{ a1} \implies \text{ A1 b0 c0 a1} \\
\text{A0} &\text{ a0} \implies \text{ A0} \\
\text{A0} &\text{ a1} \implies \text{ A1} \\
\text{a1} &\implies \text{ a0} \\
\text{A1} &\implies \text{ A0}
\end{align*}
\]

Then by removing all labels it suffices to prove relative termination of

\[
\begin{align*}
\text{acb} &\implies \text{ cbaba} \\
\text{b} &\implies \text{ cc} \\
\text{aa} &\implies \text{ abca} \\
\text{Aa} &\implies \text{ Abca} \\
\text{Aa} &\implies \text{ A}
\end{align*}
\]

21
By again finding a quasi-model labelling, applying polynomial interpretations and removing all labels the rule \( Aa \rightarrow Abca \) is removed:

Apply labelling with the following interpretation in \( \{0,1\} \):

- \( b \): constant 0
- \( c \): constant 0
- \( a \): constant 1
- \( d \): identity
- \( B \): identity
- \( A \): identity

and label every symbol by the value of its argument.

This is a quasi-model for \( 1 > 0 \).

Labelled system:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a0 )</td>
<td>( c0 b0 \rightarrow c0 b1 a0 b1 a0 )</td>
</tr>
<tr>
<td>( a0 c0 b1 \rightarrow c0 b1 a0 b1 a1 )</td>
<td></td>
</tr>
<tr>
<td>( b0 \rightarrow c0 c0 )</td>
<td></td>
</tr>
<tr>
<td>( b1 \rightarrow c0 c1 )</td>
<td></td>
</tr>
<tr>
<td>( a1 a0 \rightarrow a0 b0 c1 a0 )</td>
<td></td>
</tr>
<tr>
<td>( a1 a1 \rightarrow a0 b0 c1 a1 )</td>
<td></td>
</tr>
<tr>
<td>( A1 a0 \rightarrow A0 b0 c1 a0 )</td>
<td></td>
</tr>
<tr>
<td>( A1 a1 \rightarrow A0 b0 c1 a1 )</td>
<td></td>
</tr>
<tr>
<td>( A1 a0 \rightarrow A0 )</td>
<td></td>
</tr>
<tr>
<td>( A1 a1 \rightarrow A1 )</td>
<td></td>
</tr>
<tr>
<td>( b1 \rightarrow b0 )</td>
<td></td>
</tr>
<tr>
<td>( c1 \rightarrow c0 )</td>
<td></td>
</tr>
<tr>
<td>( a1 \rightarrow a0 )</td>
<td></td>
</tr>
<tr>
<td>( d1 \rightarrow d0 )</td>
<td></td>
</tr>
<tr>
<td>( B1 \rightarrow B0 )</td>
<td></td>
</tr>
<tr>
<td>( A1 \rightarrow A0 )</td>
<td></td>
</tr>
</tbody>
</table>

Choose polynomial interpretation \( d1 \) : lambda \( x.x+1 \), rest identity

remove: \( d1 \rightarrow d0 \)

Choose polynomial interpretation \( B1 \) : lambda \( x.x+1 \), rest identity

remove: \( B1 \rightarrow B0 \)

Choose polynomial interpretation \( A1 \) : lambda \( x.x+1 \), rest identity

remove: \( A1 a0 \rightarrow A0 b0 c1 a0 \)

remove: \( A1 a1 \rightarrow A0 b0 c1 a1 \)

remove: \( A1 a0 \rightarrow A0 \)

remove: \( A1 \rightarrow A0 \)
Remaining rules:

\[
\begin{align*}
& a_0 \ c_0 \ b_0 \rightarrow= c_0 \ b_1 \ a_0 \ b_1 \ a_0 \\
& a_0 \ c_0 \ b_1 \rightarrow= c_0 \ b_1 \ a_0 \ b_1 \ a_1 \\
& b_0 \rightarrow= c_0 \ c_0 \\
& b_1 \rightarrow= c_0 \ c_1 \\
& a_1 \ a_0 \rightarrow= a_0 \ b_0 \ c_1 \ a_0 \\
& a_1 \ a_1 \rightarrow= a_0 \ b_0 \ c_1 \ a_1 \\
& A_1 \ a_1 \rightarrow= A_1 \\
& b_1 \rightarrow= b_0 \\
& c_1 \rightarrow= c_0 \\
& a_1 \rightarrow= a_0 \\
\end{align*}
\]

Remove all labels, remaining unlabelled system:

\[
\begin{align*}
& a_0 \rightarrow= b_0 \\
& b_0 \rightarrow= c_0 \\
& c_1 \rightarrow= c_0 \\
& a_1 \rightarrow= a_0 \\
\end{align*}
\]

Next lhs's and rhs's are reversed, and for the resulting SRS for the third time in this proof a quasi-model labelling is found. Finally by consecutive polynomial interpretations all strict rules are removed, proving relative termination and concluding the termination proof of the original SRS:

Reverse every lhs and rhs of the system.

Apply labelling with the following interpretation in \{0,1\}:

\[
\begin{align*}
& b: \text{identity} \\
& c: \text{constant } 0 \\
& a: \text{identity} \\
& d: \text{constant } 0 \\
& B: \text{constant } 0 \\
& A: \text{constant } 1 \\
\end{align*}
\]

and label every symbol by the value of its argument.

This is a quasi-model for \(1 > 0\).

Labelled system:

\[
\begin{align*}
& b_0 \ c_0 \ a_0 \rightarrow= a_0 \ b_0 \ a_0 \ b_0 \ c_0 \\
& b_0 \ c_1 \ a_1 \rightarrow= a_0 \ b_0 \ a_0 \ b_0 \ c_1 \\
& b_0 \rightarrow= c_0 \ c_0 \\
& b_1 \rightarrow= c_0 \ c_1 \\
& a_0 \ a_0 \rightarrow= a_0 \ c_0 \ b_0 \ a_0 \\
& a_1 \ a_1 \rightarrow= a_0 \ c_1 \ b_1 \ a_1 \\
& a_1 \ A_0 \rightarrow= A_0 \\
& a_1 \ A_1 \rightarrow= A_1 \\
& b_1 \rightarrow= b_1 \\
& c_1 \rightarrow= c_0 \\
& a_1 \rightarrow= a_0 \\
& d_1 \rightarrow= d_0 \\
& B_1 \rightarrow= B_0 \\
& A_1 \rightarrow= A_0 \\
\end{align*}
\]
Choose polynomial interpretation  $a_1 : \text{lambda } x. x+1$, rest identity
remove: $b_0 \ C_1 \ a_1 \ \rightarrow= \ a_0 \ b_0 \ a_0 \ b_0 \ c_1$
remove: $a_1 \ a_1 \ \rightarrow= \ a_0 \ c_1 \ b_1 \ a_1$
remove: $a_1 \ A_0 \ \rightarrow= \ A_0$
remove: $a_1 \ A_1 \ \rightarrow= \ A_1$
remove: $a_1 \ \rightarrow= \ a_0$

Choose polynomial interpretation  $b_1 : \text{lambda } x. x+1$, rest identity
remove: $b_1 \ \rightarrow= \ c_0 \ c_1$
remove: $b_1 \ \rightarrow= \ b_0$

Choose polynomial interpretation  $c_1 : \text{lambda } x. x+1$, rest identity
remove: $c_1 \ \rightarrow= \ c_0$

Choose polynomial interpretation  $d_1 : \text{lambda } x. x+1$, rest identity
remove: $d_1 \ \rightarrow= \ d_0$

Choose polynomial interpretation  $B_1 : \text{lambda } x. x+1$, rest identity
remove: $B_1 \ \rightarrow= \ B_0$

Choose polynomial interpretation  $A_1 : \text{lambda } x. x+1$, rest identity
remove: $A_1 \ \rightarrow= \ A_0$

Relatively terminating since no strict rules remain.

The full complicated proof was found completely automatically by TORPA in one or two seconds.

7 Collatz problem

In this section we will show how the well-known Collatz problem can be coded as the termination problem of a SRS consisting of 7 small rules over 5 symbols. The reason to include this in this paper is twofold: on the one hand it illustrates the power of termination of small SRSs, and on the other hand techniques developed in this paper are helpful in the proof that our SRS coding is indeed correct: we even use TORPA in this proof of correctness.

The Collatz problem is called after L. Collatz who posed this problem in 1937; it is also called Syracuse problem or $3n+1$ problem. The problem is whether for every positive natural number $a_0$ there exists $n$ such that $a_n = 1$, where

$$a_n = \begin{cases} a_{n-1}/2 & \text{if } a_{n-1} \text{ is even} \\ 3a_{n-1} + 1 & \text{if } a_{n-1} \text{ is odd} \end{cases}$$

for $n = 1, 2, 3, \ldots$. It is conjectured that this holds for every positive natural number $a_0$; it has been proven in Mathematica that it holds for all positive natural numbers $< 10^{15}$ ([13]). An overview on this problem is given in [11]; much more references may be found by searching for ‘Syracuse’ or ‘Collatz’ on the internet. At a first glance one may expect that a full programming framework including integer arithmetic is required to express this conjecture. However, this is not true. Let $C$ be the SRS consisting of the following seven
The rules:
\[ h11 \rightarrow 1h \quad 11hb \rightarrow 11sb \quad h1b \rightarrow t11b \]
\[ 1s \rightarrow s1 \quad 1t \rightarrow t111 \]
\[ bs \rightarrow bh \quad bt \rightarrow bh \]

over the five symbols \( b \) (blank), \( h \) (half), \( s \) (shift), \( t \) (triple) and 1. This SRS can be seen as a kind of a Turing machine with an elastic tape: the tape alphabet consists of 1 and \( b \) where \( b \) is the blank symbol, and \( h, s \) and \( t \) are the machine states. By \( h \) the head is shifted to the right while contracting two tape cells to one; by \( s \) and \( t \) the head is shifted to the left while by \( t \) every cell is tripled.

We will show that the above conjecture is equivalent to termination of \( C \). We do not make progress in proving the conjecture (we agree with Paul Erdős that “mathematics is not yet ready for such problems”), we only prove this equivalence. Hence TORPA will not be able to prove termination of \( C \).

**Theorem 8** The SRS \( C \) is terminating if and only for every positive natural number \( a_0 \) there exists \( n \) such that \( a_n = 1 \).

**Proof:** With respect to \( C \) for \( n > 1 \) we have
\[
bh1^{2n}b \rightarrow^* b1^nhb \rightarrow b1^n sb \rightarrow^* bs1^n b \rightarrow bh1^n h
\]
and for \( n \geq 0 \) we have
\[
bh1^{2n+1}b \rightarrow^* b1^n h1b \rightarrow b1^n tl1b \rightarrow^* bt1^{3n+2} b \rightarrow bh1^{3n+2} b.
\]

This latter reduction describes two steps in the sequence: if \( a_k = 2n + 1 \) then \( a_{k+1} = 3(2n + 1) + 1 = 6n + 4 \) and \( a_{k+2} = 3n + 2 \). Hence if the number 1 does not occur in the infinite sequence \( a_0, a_1, a_2, \ldots \) then also 2 and 4 do not occur, and by the above reduction patterns we obtain an infinite \( C \)-reduction starting from the string \( bhl^{a_0} b \). This proves the ‘only if’-part of the theorem.

For the ‘if’-part we assume that the SRS admits an infinite reduction; we will show that then a number \( a_0 \) exists such that the infinite sequence \( a_0, a_1, a_2, \ldots \) does not contain the number 1, concluding the proof the theorem.

By adding symbols \( b \) in front and behind the first string in the infinite reduction, this first string is of the shape \( bu_1 bu_2 b \cdots bu_kb \) where \( u_1, \ldots, u_k \) are strings not containing \( b \). Due to the shape of the rules the only way such a string can be rewritten is that \( u_i \) is replaced by \( u'_i \) for some \( i \) satisfying \( bu_i b \rightarrow bu'_i b \), and \( u_j \) is unchanged for all \( j \neq i \). Since the number of \( b \)-s never changes during the infinite reduction and every reduction step takes place in one of the finite number of \( k \) positions, for some \( i \) there is an infinite reduction starting in the string \( bu_i b \). Omitting the index \( i \) now we have an infinite reduction starting in the string \( bab \) where \( u \) is a string not containing the symbol \( b \). Now we will prove that \( u \) contains exactly one shift symbol, where the symbols \( h, s, t \) are called shift symbols. If \( u \) does not contain shift symbols then \( bab \) is a normal form not admitting an infinite reduction.
Assume $u$ contains two or more shift symbols. In the elastic Turing machine interpretation this can be seen as a configuration with more than one head on the same tape. We will show that then $bub$ does not admit an infinite reduction. In the style of TORPA we choose a labelling in $\{0, 1\}$: we choose
\[
f_h(x) = x, \quad f_b(x) = 0, \quad f_h(x) = f_s(x) = f_t(x) = 1
\]
for $x \in \{0, 1\}$. Clearly the model requirement $f_l = f_r$ holds for all rules $l \rightarrow r$. In our argument labelling of 1 and $b$ does not play a role, hence we will only label $h, s, t$. From the shape of the rules we see that the number of shift symbols does not change during rewriting, hence this number remains always two or more in the reduction of $bub$. Since $f_h, f_s$ and $f_t$ are all constant 1, all labels left from the rightmost shift symbol will be 1. Hence in the labelled version of the infinite C-reduction of $bub$ the rules $bs_i \rightarrow bh_i$ and $bt_i \rightarrow bh_i$ are only applied for $i = 1$. So we obtain an infinite reduction of the labelled system
\[
\begin{align*}
    h_011 & \rightarrow 1h_0 & 11h_0b & \rightarrow 11s_0b & h_01b & \rightarrow t_011b \\
    h_111 & \rightarrow 1h_1 & 1s_0 & \rightarrow s_01 & 1t_0 & \rightarrow t_0111 \\
    & & 1s_1 & \rightarrow s_11 & 1t_1 & \rightarrow t_1111 \\
    & & bs_1 & \rightarrow bh_1 & bt_1 & \rightarrow bh_1
\end{align*}
\]
However, this SRS is terminating. Writing $h, s, t, H, S, T$ for $h_0, s_0, t_0, h_1, s_1, t_1$, respectively, this is easily proved by TORPA only using polynomial interpretations:

Choose polynomial interpretation
$h$: $\lambda x. x+1$, rest identity
remove: $11hb \rightarrow 11sb$
remove: $h1b \rightarrow th1b$
Choose polynomial interpretation
$S$: $\lambda x. x+1$, rest identity
remove: $bS \rightarrow bh$
Choose polynomial interpretation
$T$: $\lambda x. x+1$, rest identity
remove: $bT \rightarrow bh$
Reverse every lhs and rhs of the system and choose polynomial interpretation:
$t$ and $T$: $\lambda x. 10x$, rest $\lambda x. x+1$
remove: $h11 \rightarrow 1h$
remove: $H11 \rightarrow 1H$
remove: $1t \rightarrow t111$
remove: $1T \rightarrow T111$
Choose polynomial interpretation:
$1$: $\lambda x. 10x$, rest $\lambda x. x+1$
remove: $1s \rightarrow s1$
remove: $1S \rightarrow S1$
Terminating since no rules remain.

As a consequence, we conclude that $u$ contains exactly one shift symbol. Hence $u = bl^k\alpha l^m b$ for some $\alpha \in \{h, s, t\}$ and natural numbers $k, m$. If $\alpha = h$ then we have $bh_1l^{k+2m}b \rightarrow^* bh_1b = u$. If $\alpha = s$ then $u = bl^kS l^m b \rightarrow^* bh_1l^{k+m}b$ is the only reduction of length $k + 1$ of $u$, hence $bh_1l^{k+m}b$ occurs in the infinite reduction of $u$. If $\alpha = t$
then \( u = b^{k+1}t_{1}^{m}b \rightarrow^{*} bh^{1^{k+3}_{m}}b \) is the only reduction of length \( k + 1 \) of \( u \), hence \( bh^{1^{k+3}_{m}}b \) occurs in the infinite reduction of \( u \). In all cases we found an infinite reduction of a term of the shape \( bh^{1^{n}}b \) for some \( n \). Now choose \( a_{0} = n \). Note that reducing \( bh^{1^{n}}b \) is deterministic: in every term at most one redex occurs. Since \( bh^{1^{2k}}b \rightarrow^{*} bh^{1^{k}}b \) for every \( k > 1 \), \( bh^{1^{2k}}b \) has no infinite reduction for \( k \leq 1 \), and \( bh^{1^{2k+1}}b \rightarrow^{*} bh^{1^{3k+2}}b \) for every \( k \geq 0 \), the infinite reduction of \( bh^{1^{n}}b \) gives rise to an infinite sequence \( a_{0}, a_{1}, a_{2}, \ldots \) not containing 2 and 4, and hence not containing 1.

The SRS \( C \) admits many variations for which Theorem 8 can be proved too. There are versions over more than five symbols for which a simpler proof can be given, but we preferred to keep \( C \) as simple as possible: 7 rules over 5 symbols, and every lhs and rhs has length at most 4. We did not succeed in improving any of these three restrictions while keeping the other two.

### 8 Conclusions and Further Research

TORPA is the first tool for automatically proving termination based on semantic labelling. It is able to find termination proofs fully automatically for many SRSs. Often a generated proof is of a shape that it is very unlikely that it was ever found by a human. This is not only due to restrictions in the implementation of TORPA: there are many small SRSs for which TORPA automatically finds a termination proof, but for which finding any human proof allowing any presently known technique seems to be a really hard job. On the other hand, the generated proofs are usually not more than a few pages of text, including many details. For people being familiar with the underlying theory as explained in this paper, verifying the generated proofs is never a hard job, at most it may be boring.

However, there are very small SRSs, like the single rule \( aabb \rightarrow bbbaaa \), for which TORPA fails to find a termination proof, and MatchBox ([7]) seems to succeed.

Roughly speaking, TORPA combines a number of transformations on SRSs all having the property that termination of the original SRS can be concluded from termination of the transformed SRS. The termination proof always ends in an empty SRS, or an SRS not containing strict rules, or an SRS for which termination is proved by recursive path order. Sometimes a great number of transformations is applied consecutively, and sometimes the intermediate SRSs are much bigger than the original SRS. For instance, applying TORPA on the SRS consisting of the three rules

\[
acb \rightarrow baba, \ b \rightarrow cac, \ aa \rightarrow abca
\]

over the three symbols \( a, b, c \) yields an intermediate SRS of 27 rules over 10 symbols after applying both the dependency pair transformation and a labelling, while the ultimate termination proof is given by systematically breaking down this big SRS in a number of steps.

Except for the very often used transformation of reversing all lhs's and rhs's all techniques used in TORPA also apply for term rewriting rather than string rewriting. Hence
a natural follow up will be a version of TORPA capable of proving termination of term rewriting.

References


