AAECC (2008) 19:27–38
DOI 10.1007/s00200-008-0060-6

Adding constants to string rewriting

René Thiemann · Hans Zantema · Jürgen Giesl · Peter Schneider-Kamp

Received: 13 September 2006 / Revised: 24 May 2007 / Published online: 30 January 2008

Abstract We consider *unary term rewriting*, i.e., term rewriting with *unary signatures* where all function symbols are either unary or constants. Terms over such signatures can be transformed into strings by just reading all symbols in the term from left to right, ignoring the optional variable. By lifting this transformation to rewrite rules, any unary term rewrite system (TRS) is transformed into a corresponding string rewrite system (SRS). We investigate which properties are preserved by this transformation. It turns out that any TRS over a unary signature is terminating if and only if the corresponding SRS is terminating. In this way tools for proving termination of string rewriting can be applied for proving termination of unary TRSs. For other rewriting

Supported by the Deutsche Forschungsgemeinschaft DFG under grant GI 274/5-1. This research was done while the first author was at the LuFG Informatik 2, RWTH Aachen.

R. Thiemann (✉) · J. Giesl · P. Schneider-Kamp
LuFG Informatik 2, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany
e-mail: thiemann@informatik.rwth-aachen.de

Present Address:
R. Thiemann
Institute of Computer Science, University of Innsbruck,
Technikerstr. 21a, 6020 Innsbruck, Austria
e-mail: rene.thiemann@uibk.ac.at

J. Giesl
e-mail: giesl@informatik.rwth-aachen.de

P. Schneider-Kamp
e-mail: psk@informatik.rwth-aachen.de

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

properties including confluence, unique normal form property, weak normalization and relative termination, we show that a similar corresponding preservation property does not hold.

Keywords Term rewriting · String rewriting · Termination · Confluence

1 Introduction

Every term over a unary signature, i.e., over only constants and unary symbols, can be transformed into a corresponding string by just reading all function symbols in the term from left to right, and ignoring the variable if the term contains a variable. By lifting this transformation $\mathcal{S}tr$ to rewrite rules, any unary TRS, i.e., a TRS over a unary signature, can be transformed into a corresponding SRS. Here both the unary symbols and the constants appear as string symbols. As an example consider the following unary TRS $\mathcal{R}$ over the function symbols f, g, h, i of arity 1 and the constants b and c (x is a variable). Its corresponding SRS is $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$.

Example 1

$$\mathcal{R} = \left\{ \begin{array}{l} \mathsf{f}(x) \to \mathsf{g}(x) \\ \mathsf{f}(x) \to \mathsf{h}(\mathsf{b}) \\ \mathsf{f}(\mathsf{c}) \to \mathsf{i}(\mathsf{c}) \end{array} \right\} \qquad \mathcal{S} = \left\{ \begin{array}{c} \mathsf{f} \to \mathsf{g} \\ \mathsf{f} \to \mathsf{h}\,\mathsf{b} \\ \mathsf{f}\,\mathsf{c} \to \mathsf{i}\,\mathsf{c} \end{array} \right\}$$

Note that for each term one can build a corresponding string, but not vice versa. Some strings like $\mathsf{h}\,\mathsf{b}\,\mathsf{g}$ and $\mathsf{f}\,\mathsf{c}\,\mathsf{g}$ have no counter-part as a term.

There is also a difference between the rewrite relations $\to_{\mathcal{R}}$ and $\to_{\mathcal{S}}$. The TRS $\mathcal{R}$ in Example 1 contains all kinds of possible rules of a unary TRS: a rule with no constants, a rule with one constant, and a rule with two constants. If one considers rules of the first kind, then there is not much difference between the rewrite relation of $\mathcal{R}$ and $\mathcal{S}$. The context (resp. substitution) for the rewrite step with $\mathcal{R}$ directly correspond to the left-context (resp. right-context) for the rewrite step with $\mathcal{S}$. For rules of the third kind there already is a small difference between the rewrite relations of $\mathcal{R}$ and $\mathcal{S}$. While it is not possible to instantiate the third rule in $\mathcal{R}$, one can still use a right-context when rewriting with $\mathcal{S}$. This is important if one deals with strings like $\mathsf{f}\,\mathsf{c}\,\mathsf{g}$ that cannot be represented as terms. However, using the second rule we detect an even larger difference between $\to_{\mathcal{R}}$ and $\to_{\mathcal{S}}$. Rewriting a term $\mathsf{f}(\mathsf{g}(\dots))$ by $\mathcal{R}$ results in $\mathsf{h}(\mathsf{b})$ and hence, the subterm $\mathsf{g}(\dots)$ is dropped. This does not happen if we consider the corresponding rewrite step $\mathsf{f}\,\mathsf{g}\dots \to_{\mathcal{S}} \mathsf{h}\,\mathsf{b}\,\mathsf{g}\dots$ with the SRS $\mathcal{S}$. Here, the subterm $\mathsf{g}\dots$ which was previously dropped remains present. In particular, $\mathsf{h}(\mathsf{b})$ is a normal form w.r.t. $\mathcal{R}$, whereas $\mathsf{h}\,\mathsf{b}\,\mathsf{g}\dots$ is not a normal form w.r.t. $\mathcal{S}$ if the string $\mathsf{g}\dots$ contains a redex.

Now we arrive at the main topic of this paper: given an arbitrary unary TRS $\mathcal{R}$, how do typical rewrite properties like termination and confluence of $\mathcal{R}$ relate to the corresponding properties of $\mathcal{S}tr(\mathcal{R})$? Due to the differences sketched above, the answers to these questions are not obvious.

For the case where the unary TRS only has symbols of arity 1 and no constants, it is well known that there is a one-to-one correspondence between the term rewrite steps and the string rewrite steps, by which all instances of the above question are trivial. This paper is an investigation of the natural extension of this relationship when constants are allowed in the TRS.

As a main result we achieve that any unary TRS $\mathcal{R}$ is terminating if and only if $\mathcal{S}tr(\mathcal{R})$ is terminating. This observation was already used in the termination prover AProVE [4] for years, while the underlying theory was not yet published. For instance, termination of the TRS consisting of the rules

$$\begin{array}{rlrl} \mathsf{double}(\mathsf{s}(x)) & \to \mathsf{s}(\mathsf{s}(\mathsf{double}(\mathsf{p}(\mathsf{s}(x))))) & \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) & \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \\ \mathsf{double}(\mathsf{0}) & \to \mathsf{0} & \mathsf{p}(\mathsf{s}(\mathsf{0})) & \to \mathsf{0} \end{array}$$

is easily proved by our transformation to SRS,[1] while without this transformation it is much harder. The reason is that there are several techniques which are only applicable (or at least much easier to automate) for string rewriting than for term rewriting. Examples for such methods include string reversal, semantic labelling [6], etc. Consequently, many termination tools offer particular termination techniques that are only applicable for SRSs and there are even tools like TORPA [7] which only work for string rewriting. By our main theorem, these techniques and tools now also become applicable for term rewriting on unary symbols and constants. For instance, termination of the SRS resulting from the double-example is easily proved by TORPA. In this way, the power of termination proving is increased.

Moreover, indirectly our main result also applies for termination of TRSs having symbols of higher arity. The reason is that termination problems can often be transformed into termination problems of *unary* TRSs by applying a suitable *argument filtering* [1,3]. For instance, consider the following TRS.

$$\begin{array}{rlrl} \mathsf{plus}(\mathsf{0}, y) & \to y & \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) & \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x))) \\ \mathsf{plus}(\mathsf{s}(x), y) & \to \mathsf{s}(\mathsf{plus}(\mathsf{p}(\mathsf{s}(x)), y)) & \mathsf{p}(\mathsf{s}(\mathsf{0})) & \to \mathsf{0} \end{array}$$

When proving termination in the dependency pair framework [3] (as implemented for example in AProVE [4]), the main proof obligation is solving the dependency pair problem consisting of the dependency pair $\mathsf{PLUS}(\mathsf{s}(x), y) \to \mathsf{PLUS}(\mathsf{p}(\mathsf{s}(x)), y)$ and the rules $\mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))$ and $\mathsf{p}(\mathsf{s}(\mathsf{0})) \to \mathsf{0}$. After applying an argument filtering that removes the second argument of PLUS, this problem can be solved by proving termination of the unary TRS

$$\mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \to \mathsf{s}(\mathsf{p}(\mathsf{s}(x))),\ \ \mathsf{p}(\mathsf{s}(\mathsf{0})) \to \mathsf{0},\ \ \mathsf{PLUS}(\mathsf{s}(x)) \to \mathsf{PLUS}(\mathsf{p}(\mathsf{s}(x))).$$

This can easily be done by our transformation to SRS, by applying string rewriting techniques to prove termination, as above.

[1] To prove termination of the resulting SRS, one can first perform string reversal:

$$\begin{array}{ll} \mathsf{s\ double} \to \mathsf{s\ p\ double\ s\ s} & \mathsf{s\ s\ p} \to \mathsf{s\ p\ s} \\ \mathsf{0\ double} \to \mathsf{0} & \mathsf{0\ s\ p} \to \mathsf{0} \end{array}$$

Now termination is easily shown by dependency pairs [1] and the embedding order.

For the properties

- unique normal form property in several variants,
- confluence and local confluence,
- weak normalization, and
- relative termination

we will show that a similar 'if and only if'-property does not hold. For the unique normal form property and its variants none of the directions holds; for the others, the property for $\mathcal{S}tr(\mathcal{R})$ implies the property for $\mathcal{R}$. For all directions we either prove the property or give a counterexample.

The paper is organized as follows. In Sect. 2 we introduce the required notions and notations, and formally define the transformation from unary terms to strings and backwards. Moreover, we present first basic results about these transformations. Then in Sect. 3 we investigate the relation between one-step reductions of strings and one-step reductions of unary terms. These results are needed to analyze the relation between strings and unary terms if one regards further properties which involve more than one reduction step. In the next sections we investigate the rewrite properties for which the 'if and only if'-property does not hold, concluded by our main result: the 'if and only if'-property for termination. We conclude in Sect. 7.

2 Transformation between terms and strings

We assume familiarity with term and string rewriting (see [2] for details). We write $\mathcal{T}(\Sigma, \mathcal{V})$ for the set of terms over the unary signature Σ and the set of variables $\mathcal{V}$. As mentioned, "unary" means that there are no symbols of arity > 1. In unary term rewriting we never need more than one variable, so without loss of generality we assume that $\mathcal{V} = \{x\}$. For $t \in \mathcal{T}(\Sigma, \mathcal{V})$ we write $[x := t]$ for the substitution replacing this variable x by t.

The set of strings over Σ is Σ^*, where ε is the empty string. We use $\ell, r, s, t, \ldots$ to denote terms, $u, v, w, \ldots$ to denote strings, $\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$ to denote constants, and $\mathsf{f}, \mathsf{g}, \mathsf{h}, \ldots$ to denote symbols of arity 1.

To convert unary terms to strings and vice versa we introduce three functions. The function $\mathcal{S}tr$ converts terms to strings by just ignoring parentheses and by eliminating all variables. For the other direction from strings to unary terms, we use two functions. The problem is that strings like $u = \mathsf{f}_1 \ldots \mathsf{f}_n\, \mathsf{a}\, \mathsf{g}_1 \ldots \mathsf{g}_m\, \mathsf{b}\, \mathsf{h}_1 \ldots \mathsf{h}_k$ with (former) constants a and b in the middle cannot be completely represented as a term. Here, the function $\mathcal{T}erm$ returns the leftmost part of the string that can be represented as a term and $\mathcal{D}rop$ delivers the part of the string that is dropped by $\mathcal{T}erm$. For example, $\mathcal{T}erm(u) = \mathsf{f}_1(\ldots(\mathsf{f}_n(\mathsf{a}))\ldots)$ and $\mathcal{D}rop(u) = \mathsf{g}_1 \ldots \mathsf{g}_m\, \mathsf{b}\, \mathsf{h}_1 \ldots \mathsf{h}_k$. Definition 2 introduces these functions formally.

Definition 2 **($\mathcal{S}tr$, $\mathcal{T}erm$, and $\mathcal{D}rop$)** The functions $\mathcal{S}tr : \mathcal{T}(\Sigma, \mathcal{V}) \to \Sigma^*$, $\mathcal{T}erm : \Sigma^* \to \mathcal{T}(\Sigma, \mathcal{V})$, and $\mathcal{D}rop : \Sigma^* \to \Sigma^*$ are defined inductively by

$$\begin{array}{rlrlrl}
\mathcal{S}tr(x) &= \varepsilon & \mathcal{T}erm(\varepsilon) &= x & \mathcal{D}rop(\varepsilon) &= \varepsilon \\
\mathcal{S}tr(\mathsf{a}) &= \mathsf{a} & \mathcal{T}erm(\mathsf{a}\, w) &= \mathsf{a} & \mathcal{D}rop(\mathsf{a}\, w) &= w \\
\mathcal{S}tr(\mathsf{f}(t)) &= \mathsf{f}\, \mathcal{S}tr(t) & \mathcal{T}erm(\mathsf{f}\, w) &= \mathsf{f}(\mathcal{T}erm(w)) & \mathcal{D}rop(\mathsf{f}\, w) &= \mathcal{D}rop(w)
\end{array}$$

for all unary symbols f of arity 1 and all constants a in Σ. The transformation $\mathcal{S}tr$ is applied on unary TRSs by defining

$$\mathcal{S}tr(\mathcal{R}) = \{\mathcal{S}tr(\ell) \to \mathcal{S}tr(r) \mid \ell \to r \in \mathcal{R}\}.$$

Before investigating the difference between the rewrite relation of a unary TRS $\mathcal{R}$ and the rewrite relation of the corresponding SRS $\mathcal{S}tr(\mathcal{R})$, we need some basic properties of the functions from Definition 2.

Lemma 3 (Properties of $\mathcal{S}tr$, $\mathcal{T}erm$, $\mathcal{D}rop$) *Let $t \in \mathcal{T}(\Sigma, \mathcal{V})$, $u, v \in \Sigma^*$.*

(1) $t = \mathcal{T}erm(\mathcal{S}tr(t))$.
(2) $u = \mathcal{S}tr(\mathcal{T}erm(u))\, \mathcal{D}rop(u)$.
(3) *If $\mathcal{V}(t) = \{x\}$, then $\mathcal{S}tr(t\sigma) = \mathcal{S}tr(t)\, \mathcal{S}tr(x\sigma)$ for all substitutions σ.*
(4) $\mathcal{T}erm(u\, v) = \mathcal{T}erm(u)\, [x := \mathcal{T}erm(v)]$.
(5) *If u contains a (former) constant, then $\mathcal{T}erm(u\, w) = \mathcal{T}erm(u)$.*
(6) *If $\mathcal{T}erm(u)$ contains a variable, then $\mathcal{D}rop(u) = \varepsilon$.*

Proof The lemma can be proved by straightforward inductions on t or u. □

3 Comparing one-step reductions

For a unary TRS $\mathcal{R}$ and its corresponding SRS $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$, our main goal is to compare the rewrite relations of $\mathcal{R}$ and $\mathcal{S}$. We start by investigating the difference between one-step reductions with $\to_{\mathcal{R}}$ and $\to_{\mathcal{S}}$ and show that every $\to_{\mathcal{R}}$-step can be simulated by a corresponding $\to_{\mathcal{S}}$-step. But for the other direction one can only guarantee that every $\to_{\mathcal{S}}$-step can be simulated by zero or one $\to_{\mathcal{R}}$-step. This is illustrated by the following diagrams:

$$\begin{tikzcd}
s \arrow[r, "\mathcal{R}"] & t \\
u \arrow[u, "\mathcal{T}erm"] \arrow[r, dotted, "\mathcal{S}"'] & v \arrow[u, dotted, "\mathcal{T}erm"']
\end{tikzcd}
\qquad
\begin{tikzcd}
u \arrow[r, "\mathcal{S}"] \arrow[d, "\mathcal{T}erm"'] & v \arrow[d, dotted, "\mathcal{T}erm"] \\
\mathcal{T}erm(u) \arrow[r, dotted, "="', "\mathcal{R}"'] & \mathcal{T}erm(v)
\end{tikzcd}$$

The existence of the solid arrows implies the existence of the dotted arrows. Here, $\to_{\mathcal{R}}^{=}$ denotes the reflexive closure of $\to_{\mathcal{R}}$ (i.e., $\to_{\mathcal{R}}^{=}$ is $\to_{\mathcal{R}} \cup =$).

To illustrate the first diagram, consider the TRS $\mathcal{R}$ of Example 1 and let $s = \mathsf{f}(\mathsf{c})$, $t = \mathsf{i}(\mathsf{c})$, and $u = \mathsf{f\,c\,f}$. Then $s \to_{\mathcal{R}} t$, $\mathcal{T}erm(u) = s$, and $u \to_{\mathcal{S}} v$ for the string $v = \mathsf{i\,c\,f}$ with $\mathcal{T}erm(v) = t$. For the second diagram, let $u = \mathsf{f\,c\,f}$ and $v = \mathsf{f\,c\,g}$. Now we have $u \to_{\mathcal{S}} v$, but $\mathcal{T}erm(u) = \mathsf{f\,c} = \mathcal{T}erm(v)$.

The following two lemmas formulate and prove the above observations.

Lemma 4 (Simulating term reductions by string reductions) *Let $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$, let $u \in \Sigma^*$ be a string that corresponds to s (i.e., $s = \mathcal{T}erm(u)$), and let $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$. If $s \to_{\mathcal{R}} t$, then there exists some $v \in \Sigma^*$ such that $u \to_{\mathcal{S}} v$ and $t = \mathcal{T}erm(v)$.*

Proof Using Lemma 3, the lemma can be proven by structural induction on s. If the reduction is below the root, then the lemma directly follows from the induction hypothesis. For a reduction at the root position, the result is easily obtained by a case analysis on the three kinds of rules. □

The above lemma shows that it is possible to simulate every reduction of a unary term s by a reduction of a corresponding string u where the left part of u corresponds to s (i.e., $\mathcal{T}erm(u) = s$). However, the right part $\mathcal{D}rop$ of the string is not involved. Hence, if we now want to simulate string reductions by term reductions we obtain a different result: only if the string reduction is in the left part of the string then we can simulate it by a corresponding term reduction. But every reduction in the right part of the string which is dropped by $\mathcal{D}rop$ will not change the corresponding unary term.

Lemma 5 (Simulating string reductions by term reductions) *Let $u, v \in \Sigma^*$, let $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$. If $u \to_{\mathcal{S}} v$, then $\mathcal{T}erm(u) \to_{\mathcal{R}}^{=} \mathcal{T}erm(v)$. Moreover, if $u \to_{\mathcal{S}} v$ is a reduction without left-context, then we have a root reduction $\mathcal{T}erm(u) \to_{\mathcal{R}} \mathcal{T}erm(v)$.*

Proof We perform induction on u. We first consider the case of a reduction without left-context $u = \mathcal{S}tr(\ell)\, w \to_{\mathcal{S}} \mathcal{S}tr(r)\, w = v$ where $\ell \to r \in \mathcal{R}$. We can use Lemma 3 (4) and (1) to conclude

$$\begin{array}{l}\mathcal{T}erm(u) = \mathcal{T}erm(\mathcal{S}tr(\ell))\,[x := \mathcal{T}erm(w)] = \ell\,[x := \mathcal{T}erm(w)] \\ \hphantom{\mathcal{T}erm(u) = \mathcal{T}erm(\mathcal{S}tr(\ell))\,[x := \mathcal{T}erm(w)] = \ell\,[x :=\;} \downarrow_{\mathcal{R}} \\ \mathcal{T}erm(v) = \mathcal{T}erm(\mathcal{S}tr(r))\,[x := \mathcal{T}erm(w)] = r\,[x := \mathcal{T}erm(w)]\end{array}$$

Next, let $u = \mathsf{a}\,w \to_{\mathcal{S}} \mathsf{a}\,w' = v$ where $w \to_{\mathcal{S}} w'$. Then obviously we obtain $\mathcal{T}erm(u) = \mathsf{a} \to_{\mathcal{R}}^{=} \mathsf{a} = \mathcal{T}erm(v)$.

Finally, let $u = \mathsf{f}\,w \to_{\mathcal{S}} \mathsf{f}\,w' = v$ where $w \to_{\mathcal{S}} w'$. By the induction hypothesis, we get $\mathcal{T}erm(u) = \mathsf{f}(\mathcal{T}erm(w)) \to_{\mathcal{R}}^{=} \mathsf{f}(\mathcal{T}erm(w')) = \mathcal{T}erm(v)$. □

Based on the above observations on the relationship between one-step reductions of strings and of terms, in the remaining sections we will investigate further properties which involve more than one reduction step.

4 Unique normal form property

In this section we analyze some variants of the unique normal form property. Following [5] we consider the following variants, all defined for an arbitrary abstract reduction relation $\to$:

- *normal form property* (NF):
  if $t \leftrightarrow^* u$ and u is a normal form, then $t \to^* u$;
- *unique normal form property* (UN):
  if $t \leftrightarrow^* u$ and t, u are normal forms, then $t = u$;
- *unique normal form property with respect to reduction* (UN$^{\to}$):
  if $t \leftarrow^* \cdot \to^* u$ and t, u are normal forms, then $t = u$.

So $\mathrm{UN}^{\rightarrow}$ means that every term has at most one normal form. The implications NF $\Rightarrow$ UN $\Rightarrow$ $\mathrm{UN}^{\rightarrow}$ are immediate from this definition.

We saw in Sect. 3 that in string reductions, we might obtain additional suffixes in the strings which are not present in the corresponding terms. Thus, if an SRS produces different normal forms where the difference is only in the suffix part of a string, then it could be that as terms there is no difference anymore and we have unique normal forms.

On the other hand, one could have different normal forms as terms but not with strings: Consider strings with different prefixes and non-normalizing suffixes. The suffixes prohibit normalization and hence these strings have no normal forms at all. However, if we regarded the strings as terms, we would only look at the different prefixes which may be normal forms.

The following theorem shows that both these situations can arise, for all three variants of the normal form property.

Theorem 6 (Unique normal form property)

(1) *There is a unary TRS $\mathcal{R}$ satisfying all three properties* NF, UN, *and* $\mathrm{UN}^{\rightarrow}$*, while $\mathcal{S}tr(\mathcal{R})$ satisfies none of these three properties.*

(2) *There is a unary TRS $\mathcal{R}$ satisfying none of the three properties* NF, UN, *and* $\mathrm{UN}^{\rightarrow}$*, while $\mathcal{S}tr(\mathcal{R})$ satisfies all of these three properties.*

Proof (1) Define $\mathcal{R}$ and $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$ by:

$$\mathcal{R} = \left\{ \begin{array}{l} \mathsf{f(b)} \to \mathsf{f(a)} \\ \mathsf{f}(x) \to \mathsf{a} \end{array} \right\} \qquad \mathcal{S} = \left\{ \begin{array}{c} \mathsf{f\,b} \to \mathsf{f\,a} \\ \mathsf{f} \to \mathsf{a} \end{array} \right\}$$

$\mathcal{R}$ is confluent since it is terminating and locally confluent. The reason for the latter is that the only critical pair consisting of $\mathsf{f(a)}$ and a is joinable. Then from confluence we can conclude all three properties NF, UN, and $\mathrm{UN}^{\rightarrow}$.

However in $\mathcal{S}$, the string $\mathsf{f\,b}$ has two normal forms: $\mathsf{f\,b} \to_{\mathcal{S}} \mathsf{f\,a} \to_{\mathcal{S}} \mathsf{a\,a}$ and $\mathsf{f\,b} \to_{\mathcal{S}} \mathsf{a\,b}$. Hence $\mathcal{S}$ does not satisfy $\mathrm{UN}^{\rightarrow}$. By NF $\Rightarrow$ UN $\Rightarrow$ $\mathrm{UN}^{\rightarrow}$, we conclude that $\mathcal{S}$ neither satisfies UN, nor NF.

(2) Define $\mathcal{R}$ and $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$ by:

$$\mathcal{R} = \left\{ \begin{array}{r} \mathsf{a} \to \mathsf{f(a)} \\ \mathsf{a} \to \mathsf{g(a)} \\ \mathsf{f}(x) \to \mathsf{b} \quad \\ \mathsf{g}(x) \to \mathsf{c} \quad \end{array} \right\} \qquad \mathcal{S} = \left\{ \begin{array}{l} \mathsf{a} \to \mathsf{f\,a} \\ \mathsf{a} \to \mathsf{g\,a} \\ \ \mathsf{f} \to \mathsf{b} \\ \mathsf{g} \to \mathsf{c} \end{array} \right\}$$

$\mathcal{R}$ does not satisfy $\mathrm{UN}^{\rightarrow}$ as a reduces to two distinct normal forms b and c. By NF $\Rightarrow$ UN $\Rightarrow$ $\mathrm{UN}^{\rightarrow}$, $\mathcal{R}$ neither satisfies UN, nor NF.

However, for $\mathcal{S}$ the situation is different. Let v be a normal form in $\mathcal{S}$ and $u \leftrightarrow^*_{\mathcal{S}} v$. As v is a normal form, it does not contain the symbol a. From the shape of the rules we conclude that only rules of the SRS $\mathcal{S}' = \{\mathsf{f} \to \mathsf{b}, \mathsf{g} \to \mathsf{c}\}$ are involved in this conversion, i.e. $u \leftrightarrow^*_{\mathcal{S}'} v$. Since $\mathcal{S}'$ is orthogonal and thereby confluent, we know that $\mathcal{S}'$ satisfies NF. Thus, $u \to^*_{\mathcal{S}'} v$ must hold. As $\mathcal{S}' \subseteq \mathcal{S}$ we obtain the reduction $u \to_{\mathcal{S}}^* v$ which proves that $\mathcal{S}$ satisfies NF. The other two properties follow from NF $\Rightarrow$ UN $\Rightarrow$ UN$^{\to}$. □

5 Confluence

A stronger property than NF, UN, and UN$^{\to}$ is confluence. Here, *every* reduction starting from the same term must be joinable (not only normalizing reductions). The counterexample in the proof of Theorem 6 (1) in the previous section already shows why confluence of a TRS does not imply confluence of the corresponding SRS: there may be differences in the suffixes of the resulting strings, whereas these different suffixes are dropped in the terms. Therefore, we will not obtain a positive result for this direction.

However, our other counterexample from the previous section where non-termination was the reason for unique normal forms will not work for confluence. For confluence, reductions must be joinable regardless of normalization. Indeed, confluence of an SRS implies confluence of the corresponding TRS. To give some intuition on this result, consider what is necessary to obtain confluence of a string u that can be reduced to v_1 and v_2. We must be able to join the strings v_1 and v_2 where we have to join both the prefixes and the suffixes of v_1 and v_2. However, regarded as terms we only have to join the prefixes which can be a strictly weaker requirement. The relationship between confluence of TRSs and SRSs is stated and proved in Theorem 7.

Theorem 7 (Confluence) *Let $\mathcal{R}$ be a unary TRS and let $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$ be the corresponding SRS.*

(1) *Confluence of $\mathcal{S}$ implies confluence of $\mathcal{R}$.*
(2) *Local confluence of $\mathcal{S}$ implies local confluence of $\mathcal{R}$.*
(3) *Confluence of $\mathcal{R}$ does not even imply local confluence of $\mathcal{S}$.*

Proof

(1) Let $s \to_{\mathcal{R}}^* t_1$ and $s \to_{\mathcal{R}}^* t_2$. We define $u = \mathcal{S}tr(s)$. Then by Lemma 3 (1) we conclude $\mathcal{T}erm(u) = \mathcal{T}erm(\mathcal{S}tr(s)) = s$. Thus, by Lemma 4 there exist strings v_i with $u \to_{\mathcal{S}}^* v_i$ where $t_i = \mathcal{T}erm(v_i)$ for $i \in \{1, 2\}$. By confluence of $\mathcal{S}$ we know that $v_i \to_{\mathcal{S}}^* w$ for some string w. Lemma 5 finally implies joinability of t_1 and t_2, as $t_i = \mathcal{T}erm(v_i) \to_{\mathcal{R}}^* \mathcal{T}erm(w)$.
(2) The statement is proved as in (1).
(3) We can take the same counterexample as in the proof of Theorem 6 (1):

$$\mathcal{R} = \left\{ \begin{array}{l} \mathsf{f(b)} \to \mathsf{f(a)} \\ \mathsf{f}(x) \to \mathsf{a} \end{array} \right\} \qquad \mathcal{S} = \left\{ \begin{array}{c} \mathsf{f\,b} \to \mathsf{f\,a} \\ \mathsf{f} \to \mathsf{a} \end{array} \right\}$$

The SRS $\mathcal{S}$ is not locally confluent. The string $\mathsf{f\,b}$ can be reduced in one step to either $\mathsf{f\,a}$ or to the normal form $\mathsf{a\,b}$. However, there is obviously no possibility to reduce $\mathsf{f\,a}$ to $\mathsf{a\,b}$ as the only possible reduction of $\mathsf{f\,a}$ yields the normal form $\mathsf{a\,a}$. Nevertheless, $\mathcal{R}$ is confluent. As $\mathcal{R}$ is terminating, we only have to show local confluence by joinability of critical pairs. The only critical pair of $\mathcal{R}$ is $(\mathsf{f(a)}, \mathsf{a})$ which is joinable by reducing $\mathsf{f(a)}$ to a by the second rule. □

6 Normalization

In this section we investigate properties related to normalization: weak normalization, relative termination, and termination.

For weak normalization, i.e., the property that every term or string has a normal form, we obtain the same results as for confluence: Weak normalization of an SRS implies weak normalization of the corresponding TRS but not vice versa. The reason is again that the suffix of a string can contain a non-normalizing part which is not present in the corresponding term.

Theorem 8 (Weak normalization) *Let $\mathcal{R}$ be a TRS and $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$ be the corresponding SRS.*

(1) *Weak normalization of $\mathcal{S}$ implies weak normalization of $\mathcal{R}$.*
(2) *Weak normalization of $\mathcal{R}$ does not imply weak normalization of $\mathcal{S}$.*

Proof

(1) Let s be a term. We define $u = \mathcal{S}tr(s)$ and get $\mathcal{T}erm(u) = \mathcal{T}erm(\mathcal{S}tr(s)) = s$ by Lemma 3 (1). As $\mathcal{S}$ is weakly normalizing, $u \to_{\mathcal{S}}^{*} w$ for some $\mathcal{S}$-normal form w, By Lemma 5, $s = \mathcal{T}erm(u) \to_{\mathcal{R}}^{*} \mathcal{T}erm(w)$. Note that $\mathcal{T}erm(w)$ must be an $\mathcal{R}$-normal form (i.e., s has a normal form, which proves weak normalization of $\mathcal{R}$). Otherwise by Lemma 4, w could be reduced further by $\to_{\mathcal{S}}$. This contradicts the fact that w is an $\mathcal{S}$-normal form.
(2) Consider the following TRS and its corresponding SRS:

$$\mathcal{R} = \left\{ \begin{array}{r} \mathsf{a} \to \mathsf{f}(\mathsf{a}) \\ \mathsf{f}(x) \to \mathsf{b} \end{array} \right\} \qquad \mathcal{S} = \left\{ \begin{array}{c} \mathsf{a} \to \mathsf{f}\,\mathsf{a} \\ \mathsf{f} \to \mathsf{b} \end{array} \right\}$$

$\mathcal{R}$ is weakly normalizing: All (sub)terms with root f or a can be reduced to b. In this way, we obtain a term which does neither contain f nor a. Obviously, any such term must be a normal form.
However, a is not weakly normalizing in $\mathcal{S}$, since $\to_{\mathcal{S}}$ cannot delete occurrences of a. But a string containing a is not in $\mathcal{S}$-normal form. □

A rewrite system $\mathcal{R}$ *terminates relative* to $\mathcal{R}'$ if every $\mathcal{R} \cup \mathcal{R}'$-reduction contains only finitely many $\mathcal{R}$-steps. For relative termination we get a similar result as for confluence and weak normalization: the property for string rewriting implies the property for unary term rewriting, but not vice versa.

Theorem 9 (Relative termination) *Let $\mathcal{R}$ and $\mathcal{R}'$ be unary TRSs and let $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$ and $\mathcal{S}' = \mathcal{S}tr(\mathcal{R}')$ be the corresponding SRSs.*

(1) *Termination of $\mathcal{S}$ relative to $\mathcal{S}'$ implies termination of $\mathcal{R}$ relative to $\mathcal{R}'$.*
(2) *Termination of $\mathcal{R}$ relative to $\mathcal{R}'$ does not imply termination of $\mathcal{S}$ relative to $\mathcal{S}'$.*

Proof

(1) Let $\mathcal{S}$ be terminating relative to $\mathcal{S}'$, but assume that $\mathcal{R}$ is not terminating relative to $\mathcal{R}'$. Then there is an infinite $\mathcal{R} \cup \mathcal{R}'$-reduction $t_1 \to t_2 \to \dots$ containing infinitely many $\mathcal{R}$-steps. By Lemma 3 (1) we know that $u_1 = \mathcal{S}tr(t_1)$ satisfies $\mathcal{T}erm(u_1) = \mathcal{T}erm(\mathcal{S}tr(t_1)) = t_1$. We apply Lemma 4 to obtain $u_i \to_{\mathcal{S}\cup\mathcal{S}'} u_{i+1}$ where $\mathcal{T}erm(u_i) = t_i$ for all i. Here we have $u_i \to_{\mathcal{S}} u_{i+1}$ if $t_i \to_{\mathcal{R}} t_{i+1}$, and $u_i \to_{\mathcal{S}'} u_{i+1}$ if $t_i \to_{\mathcal{R}'} t_{i+1}$. As a consequence we have an infinite $\mathcal{S} \cup \mathcal{S}'$-reduction $u_1 \to u_2 \to \dots$ containing infinitely many $\mathcal{S}$-steps, contradicting the termination of $\mathcal{S}$ relative to $\mathcal{S}'$.

(2) Consider the following TRSs and their corresponding SRSs:

$$\begin{array}{rlrl} \mathcal{R} &= \{\mathsf{f}(x) \to \mathsf{b}\} & \qquad \mathcal{S} &= \{\mathsf{f} \to \mathsf{b}\} \\ \mathcal{R}' &= \{\mathsf{a} \to \mathsf{f}(\mathsf{a})\} & \qquad \mathcal{S}' &= \{\mathsf{a} \to \mathsf{f\,a}\} \end{array}$$

$\mathcal{R}$ is terminating relative to $\mathcal{R}'$ since each application of the $\mathcal{R}$-rule reduces the number of a-symbols. Hence, after finitely many $\mathcal{R}$-steps, the resulting term contains no a and thus, the $\mathcal{R}'$-rule is not applicable anymore. However, $\mathcal{R}$ on its own is obviously terminating.

But $\mathcal{S}$ is not terminating relative to $\mathcal{S}'$ due to the following reduction:

$$\mathsf{a} \to_{\mathcal{S}'} \mathsf{f\,a} \to_{\mathcal{S}} \mathsf{b\,a} \to_{\mathcal{S}'} \mathsf{b\,f\,a} \to_{\mathcal{S}} \mathsf{b\,b\,a} \to_{\mathcal{S}'} \cdots$$

□

Next we analyze termination, also called strong normalization, i.e., the property that all reductions are finite. The result that termination of the corresponding SRS implies termination of the TRS is a special case of Theorem 9. However, in contrast to the previous theorems, here the converse direction holds as well, i.e., termination of a TRS implies termination of the corresponding SRS. To prove this observation, we use a basic theorem on *dependency pairs*. To do so, we first recapitulate some terminology from [1].

For a TRS $\mathcal{R}$ over a signature Σ a symbol $f \in \Sigma$ is called *defined* if f is the root symbol of a left-hand side of a rule of R. For every defined symbol $f \in \Sigma$ we add a fresh function symbol F (typically denoted by the corresponding upper case letter) where F and f have the same arity.

If $f(s_1, \dots, s_n) \to r$ is a rule in $\mathcal{R}$ and r contains a subterm $g(t_1, \dots, t_m)$ for a defined symbol g, then the rule $F(s_1, \dots, s_n) \to G(t_1, \dots, t_m)$ is a *dependency pair* of $\mathcal{R}$. $DP(\mathcal{R})$ denotes the TRS of all dependency pairs of $\mathcal{R}$. An infinite $\mathcal{R} \cup DP(\mathcal{R})$-reduction is called an *infinite $\mathcal{R}$-chain* if

- all $DP(\mathcal{R})$-steps are root steps, and
- the reduction contains infinitely many $DP(\mathcal{R})$-steps.

Theorem 10 (Dependency Pair Theorem [1]) *A TRS $\mathcal{R}$ is terminating if and only if no infinite $\mathcal{R}$-chain exists.*

Now we show the connection between termination of TRSs and SRSs.

Theorem 11 (Termination) *Let $\mathcal{R}$ be a unary TRS and let $\mathcal{S} = \mathcal{S}tr(\mathcal{R})$ be the corresponding SRS. Then $\mathcal{R}$ is terminating if and only if $\mathcal{S}$ is terminating.*

Proof The 'if'-part follows from part (1) of Theorem 9, where $\mathcal{R}'$ is empty.

For the 'only if'-part assume that $\mathcal{R}$ is terminating and $\mathcal{S}$ is not terminating. Then according to Theorem 10 there is an infinite $\mathcal{S}$-chain $u_1 \to u_2 \to \cdots$, where each reduction $u_i \to u_{i+1}$ is either an $\mathcal{S}$-step or a $DP(\mathcal{S})$-step without left-context. If it is an $\mathcal{S}$-step, then we have $\mathcal{T}erm(u_i) \to_{\mathcal{R}}^{=} \mathcal{T}erm(u_{i+1})$ by Lemma 5. If it is a $DP(\mathcal{S})$-step without left-context, then we have a root step $\mathcal{T}erm(u_i) \to_{DP(\mathcal{R})} \mathcal{T}erm(u_{i+1})$ by Lemma 5, using $\mathcal{S}tr(DP(\mathcal{R})) = DP(\mathcal{S})$ which is obvious from the definition. Hence by applying $\mathcal{T}erm$ to all u_i, the infinite $\mathcal{S}$-chain $u_1 \to u_2 \to \cdots$ is transformed into an infinite $\mathcal{R}$-chain. By Theorem 10, this is a contradiction to the termination of $\mathcal{R}$.

□

7 Concluding remarks

Our results can be summarized by the following table where "$\mathcal{R} \implies \mathcal{S}$" indicates whether a property of a TRS carries over to the corresponding SRS and where "$\mathcal{S} \implies \mathcal{R}$" indicates whether the converse direction holds.

Property	$\mathcal{R} \implies \mathcal{S}$	$\mathcal{S} \implies \mathcal{R}$
Normal form	no	no
Unique normal form	no	no
Unique normal form w.r.t. reductions	no	no
Local confluence	no	yes
Confluence	no	yes
Weak normalization	no	yes
Relative termination	no	yes
Termination	yes	yes

As the main result we consider the "if and only if" property for termination (Theorem 11). We proved this using the well-known dependency pair result of Theorem 10. It is also possible to prove Theorem 11 independent of dependency pairs by analyzing how an arbitrary string splits up into smaller substrings corresponding to unary terms, and by showing that every string rewrite step localizes to a term rewrite step that corresponds to one of these smaller substrings. Then either a multiset argument or a minimality argument has to be given. In our present proof a similar minimality argument is hidden in the proof of the dependency pair result.

References

1. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theor. Comp. Sci. **236**, 133–178 (2000)
2. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press, London (1998)
3. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: combining techniques for automated termination proofs. In: Proc. 11th LPAR, LNAI 3452, pp. 301–331 (2005)

4. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic termination proofs in the dependency pair framework. In: Proc. 3rd IJCAR, LNAI 4130, pp. 281–286 (2006). Tool: http://aprove.informatik.rwth-aachen.de/
5. Terese: Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science 55. Cambridge University Press, London (2003)
6. Zantema, H.: Termination of term rewriting by semantic labelling. Fundam. Inform. **24**, 89–105 (1995)
7. Zantema, H.: Termination of string rewriting proved automatically. J. Autom. Reason. **34**, 105–139 (2005). Tool: http://www.win.tue.nl/~hzantema/torpa.html