Automatically finding particular term rewriting systems

Hans Zantema

Department of Computer Science, TU Eindhoven, The Netherlands
Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands
`h.zantema@tue.nl`

Abstract

Our goal is to give a list of rewriting properties, and then automatically find a term rewriting system (TRS) satisfying these properties. In earlier work we did this for finite abstract reduction systems; in this paper we extend the approach to ground term rewriting systems over constants and one unary symbol. In particular, we fully automatically find a TRS that is locally confluent but not confluent, and for which the inverse is terminating. While for finding finite abstract reduction systems the underlying technique is boolean satisfiability (SAT), here we exploit its extension to satisfiability modulo theories (SMT), for the theory of linear inequalities.

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

It is well-known that local confluence does not imply confluence; the simplest example is the rewriting system on four constants a, b, c, d having the four rules $a \to b, b \to a, a \to c, b \to d$:

In our earlier paper [6] the main goal was to find such examples fully automatically, that is, given a combination of properties like termination, confluence and several variants, an example of a rewriting system is found over finitely many constants satisfying the given properties. We described our tool `Carpa` together with its input language for giving such a combination of properties, that automatically generates such examples by first expressing the constraints in a boolean satisfiability problem, then calling a SAT solver, and then transforming the resulting satisfying assignment back to the desired rewriting system. By entering the combination of local confluence and the negation of confluence, indeed the above example is found automatically.

However, `Carpa` looks for rewriting systems only over constants, that is, for finite abstract reduction systems (ARSs). For some combinations of properties such a finite ARS does not exist, while in a richer class of rewriting systems easily examples are found. As a simple example of this phenomenon assume we are looking for a rewriting system that is terminating, but for which its inverse is not terminating. In the world of finite ARSs such a rewriting system does not exist: if the system is terminating then it does not admit a cycle, and then also its inverse is terminating. But as soon as we allow a unary symbol in our rewriting system the situation is different: the single rewrite rule $f(a) \to a$ is terminating, while its inverse $a \to f(a)$ admits an infinite reduction.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

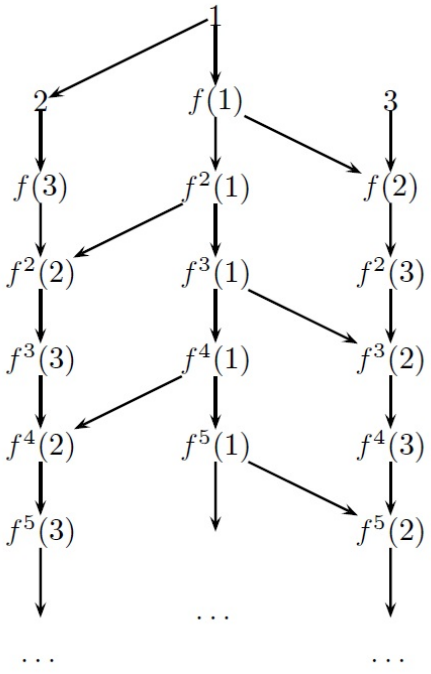
The goal of the current paper is to automatically find examples in this richer class of rewriting systems. For the underlying machinery this is a much harder job: instead of dealing with only finitely many elements, now we have infinitely many terms, and then describing properties like termination and confluence in a SAT formula is much harder. To keep it feasible we decided to restrict to a quite limited class of term rewriting systems (TRSs): we only consider ground TRSs over constants and a single unary operation symbol. On the one hand this looks quite restrictive, on the other hand it is rich enough to cover the behavior just sketched: this class covers the the single rewrite rule $f(a) \to a$ which is terminating but for which its inverse is not. We even make a further restriction: in the basic TRSs we are looking for we only allow rules of the shape $a \to b$, $f(a) \to b$ and $a \to f(b)$, where a, b are constants and f is the unary symbol.

Our leading example is finding a TRS which is locally confluent but not confluent, and for which the inverse is terminating. In the world of finite ARSs this does not have a solution: if for such a finite ARS the inverse is terminating then the system itself is terminating, and then by Newman's Lemma local confluence implies confluence, see e.g. [1, 5]. But our prototype tool `Carpa+` available from `http://www.win.tue.nl/~hzantema/carpa.html`, based on the techniques of this paper finds the following example satisfying the given properties fully automatically:

```
1 -> 2
1 -> f(1)
2 -> f(3)
3 -> f(2)
```

In fact, the above text is the literal output of `Carpa+`.

Indeed, this TRS is locally confluent as is easily checked by finding the common reduct $f(f(2))$ for the only critical pair $[2, f(1)]$. Its inverse is terminating since this inverse consists of the single rule $2 \to 1$ and three rules all removing a symbol f. The TRS has the following reduction graph:



Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 2–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

From this reduction graph we easily see that it is not confluent: 1 rewrites both to 2 in the left column and to $f(2)$ in the right column, while 2 and $f(2)$ have no common reduct since all reducts of 2 are in the left column and all reducts of $f(2)$ are in the right column.

For main properties like confluence and local confluence we did not succeed in describing them exactly, in the sense of creating a formula that is satisfiable if and only if the desired property holds. For ground TRSs it is known that confluence is decidable, see e.g. [2]. However, for encoding confluence in satisfiability the corresponding algorithm is not suitable. Instead for the confluence related properties we deal with approximations. A main problem is to describe the relation $\to_R^*$: this can typically not be described as a single step with respect to a finite TRS. For instance, if R contains the rule $f(a) \to a$, then $f^n(a) \to_R^* a$ for every n. In our approach we represent a subset $\to_R^*$ by finitely many rewrite rules, and use this to approximate $\to_R^*$ in the definition of local confluence. In this way we succeed in describing a property being slightly stronger than local confluence, and if local confluence is required instead we require this slightly stronger property, by which the approach remains sound.

For requiring non-confluence we need an approximation in the other direction. Here we project a TRS to a finite ARS by identifying $f^k(a)$ and $f^n(a)$ for all k, n that are equal modulo m for some given number m, typically $m = 2$. We prove that confluence is preserved by projection, so for instead of requiring non-confluence it is sound to require non-confluence of the finite ARS obtained by projection, and then for describing non-confluence of this finite ARS we exploit the techniques of [6].

For termination we succeed in giving an exact description: for a ground TRS over constants and a single unary symbol we succeed in finding a formula that is satisfiable if and only if the TRS is terminating. This formula is not a propositional formula in boolean variables, but involves real valued variables and linear inequalities among them. So instead of SAT we now need SMT (satisfiability modulo theories), for the theory consisting of linear inequalities. Fortunately, current tools like `yices` easily deal with this kind of SMT, so that is what we use in our approach.

This paper is organized as follows. In Section 2 we collect some standard definitions as they will be used throughout the paper. In Section 3 we present some theory on ground rewriting over constants and a single unary symbol, in particular for subTRSs of T_1 and T_2, being the TRSs of all rules of a particular shape, and how composition can be approximated. In Section 4 this theory is applied to approximate the notion of local confluence. In Section 5 we present our exact characterization of termination by means of an SMT formula. In Section 6 we describe how to project a TRS to a finite ARS and show that this preserves confluence. In Section 7 we describe our prototype tool `Carpa+` in which the techniques based on this theory have been implemented, and present our leading example. In Section 8 some more examples are given. We conclude by Section 9.

2 Preliminaries

We start by recalling some basic notions that we will use throughout the paper.

A *binary relation* R on a set A is defined to be a subset of $A \times A$. For $x, y \in A$ we will use xRy as an abbreviation of $(x, y) \in R$.

A binary relation R on a set A is called *reflexive* if $\forall x \in A : xRx$ holds.

A binary relation R on a set A is called *irreflexive* if $\forall x \in A : \neg(xRx)$ holds.

A binary relation R on a set A is called *symmetric* if $\forall x, y \in A : xRy \to yRx$ holds.

A binary relation R on a set A is called *transitive* if $\forall x, y, z \in A : (xRy \wedge yRz) \to yRz$

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 3–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

holds.

For two binary relations R, S on a set A its *composition* $R \cdot S$ is defined to be the relation

$$R \cdot S \;=\; \{(x,z) \in A \times A \mid \exists y \in A : (xRy \wedge ySz)\}.$$

The *identity relation* I on A is defined by $I = \{(x,x) \mid x \in A\}$.

For $i \geq 0$ the relation R^i is defined inductively by $R^0 = I$ and $R^{i+1} = R \cdot R^i$ fro $i \geq 0$; we write R^* for $\bigcup_{i=0}^{\infty} R^i$.

For a binary relation R on a set A its *inverse* R^{-1} is defined by $R^{-1} = \{(x,y) \in A \times A \mid yRx\}$.

A binary relation R on a set A is called *terminating* if no infinite sequence $a_1, a_2, a_3, \ldots \in A$ exists such that $a_i R a_{i+1}$ for all $i > 0$.

A binary relation R on a set A is called *confluent* if $(R^*)^{-1} \cdot R^* \;\subseteq\; R^* \cdot (R^*)^{-1}$; it is called *locally confluent* if $R^{-1} \cdot R \;\subseteq\; R^* \cdot (R^*)^{-1}$.

A term rewriting system (TRS) R is called terminating, confluent or locally confluent if its rewrite relation $\to_R$ satisfies the corresponding property. For further background on term rewriting we refer to [1, 5].

3 Basic theory

We fix a number n and define $A = \{1, 2, \ldots, n\}$. We choose the signature $\Sigma = \{f\} \cup A$, where f has arity 1 and all elements of A have arity zero. Let $\mathcal{T}$ be the set of all ground terms over Σ, that is, the set of terms of the shape $f^i(a)$ for $i \geq 0$ and $a \in A$.

For $i, j \geq 0$ and $X \subseteq A \times A$ write R_{ijX} for the ground TRS over Σ consisting of the rules $f^i(a) \to f^j(b)$ for $(a,b) \in X$.

▶ **Lemma 1.** *Let $i, j \geq 0$, $X \subseteq A \times A$ and $t, u \in \mathcal{T}$. Then $t \to_{R_{ijX}} u$ if and only if $t = f^k(a)$ and $u = f^m(b)$ for $(a,b) \in X$ and $k \geq i$ and $m = k - i + j$.*

Proof. A rule $f^i(a) \to f^j(b)$ applies on a term t if and only if $t = f^k(a)$ for $k \geq i$, and the result of this rewriting is $f^{k-i+j}(b)$. ◀

▶ **Theorem 2.** *Let $i, j, k, m \geq 0$ and $X, Y \subseteq A \times A$. Then*

$$\to_{R_{ijX}} \cdot \to_{R_{kmY}} = \to_{R_{pqX \cdot Y}}$$

for $p = \max(i, i + k - j)$ and $q = \max(m, m + j - k)$.

Proof. First assume $t \to_{R_{ijX}} \cdot \to_{R_{kmY}} u$. Then a term v exists such that $t \to_{R_{ijX}} v$ and $v \to_{R_{kmY}} u$. From Lemma 1 we then conclude that $t = f^x(a)$, $v = f^y(b)$ and $u = f^z(c)$ for $(a,b) \in X$, $(b,c) \in Y$, $x \geq i$, $y = x - i + j$, $y \geq k$ and $z = y - k + m$. Using these properties we conclude $(a,c) \in X \cdot Y$. The property $x - i + j = y \geq k$ yields $x \geq i + k - j$; combined with $x \geq i$ this yields $x \geq \max(i, i + k - j) = p$.

Further, combining $y = x - i + j$ and $z = y - k + m$ yields $z = x - i - k + j + m$. By case analysis on $k \geq j$ and $j \geq k$ this yields

$$z = x - \max(i, i + k - j) + \max(m, m + j - k) = x - p + q.$$

Hence from Lemma 1 we conclude $t \to_{R_{pqX \cdot Y}} u$, proving one direction of the theorem.

Conversely, assume $t \to_{R_{pqX \cdot Y}} u$. Then $t = f^x(a)$ and $u = f^{x-p+q}(c)$ for $x \geq p$ and $(a,c) \in X \cdot Y$, according to Lemma 1. Since $(a,c) \in X \cdot Y$ there exists $b \in A$ such that $(a,b) \in X$ and $(b,c) \in Y$. Let $v = f^{x-i+j}(b)$, then since $x \geq p \geq i$ using Lemma 1 we

conclude $t \to_{R_{ijX}} v$. It remains to prove that $v \to_{R_{kmY}} u$. According to Lemma 1 this can be concluded from $x - i + j \geq k$ which follows from $x \geq p = \max(i, i + k - j)$, and from $x - i + j - k + m = z = x - p + q$ as we observed before. ◀

▶ **Lemma 3.** *Let $i, j, k \geq 0$ and $X \subseteq A \times A$. Then*

$$\to_{R_{i+k,j+k,X}} \subseteq \to_{R_{ijX}} .$$

Proof. This follows from the observation that if t rewrites to u with respect to the rule $f^{i+k}(a) \to f^{j+k}(b)$, then also t rewrites to u with respect to the rule $f^i(a) \to f^j(b)$. ◀

We define the ground TRS T_1 over Σ to consist of all rules of the shape $a \to b$, $a \to f(b)$ and $f(a) \to b$ for $a, b \in A$, so $T_1 = R_{00A^2} \cup R_{01A^2} \cup R_{10A^2}$. Note that T_1 has $3n^2$ rules. Every subTRS of T_1 can be written as

$$R(X_{00}, X_{01}, X_{10}) = R_{00X_{00}} \cup R_{01X_{01}} \cup R_{10X_{10}}$$

for three sets $X_{00}, X_{01}, X_{10} \subseteq A \times A$, so.

$$R(X_{00}, X_{01}, X_{10}) =$$

$$\{a \to b \mid (a, b) \in X_{00}\} \cup \{a \to f(b) \mid (a, b) \in X_{01}\} \cup \{f(a) \to b \mid (a, b) \in X_{10}\}.$$

As T_1 has $3n^2$ rules, there are 2^{3n^2} such subTRSs, each to be represented by $3n^2$ booleans.

Similarly, we define the ground TRS T_2 to consist of the $6n^2$ rules of the shape $a \to b$, $a \to f(b)$, $f(a) \to b$, $f(a) \to f(b)$, $a \to f(f(b))$ and $f(f(a)) \to b$, for $a, b \in A$, so $T_2 = R_{00A^2} \cup R_{01A^2} \cup R_{10A^2} \cup R_{11A^2} \cup R_{02A^2} \cup R_{20A^2}$. SubTRSs of T_2 can be written as

$$R(X_{00}, X_{01}, X_{10}, X_{11}, X_{02}, X_{20}) \;=$$

$$R_{00X_{00}} \cup R_{01X_{01}} \cup R_{10X_{10}} \cup R_{11X_{11}} \cup R_{02X_{02}} \cup R_{20X_{20}}$$

for six sets $X_{00}, X_{01}, X_{10}, X_{11}, X_{02}, X_{20} \subseteq A \times A$. Observe that $T_1 \subseteq T_2$ and $R(X_{00}, X_{01}, X_{10}) = R(X_{00}, X_{01}, X_{10}, \emptyset, \emptyset, \emptyset)$.

For two subTRSs $R = R(X_{00}, X_{01}, X_{10}, X_{11}, X_{02}, X_{20})$ and $S = R(X'_{00}, X'_{01}, X'_{10}, X'_{11}, X'_{02}, X'_{20})$ of T_2 we define $\mathsf{comp}(R, S) = R(Y_{00}, Y_{01}, Y_{10}, Y_{11}, Y_{02}, Y_{20})$ for Y_{00}, Y_{01}, Y_{10}, Y_{11}, Y_{02}, Y_{20} defined by

$$\begin{array}{rcl}
Y_{00} & = & X_{00} \cdot X'_{00} \cup X_{01} \cdot X'_{10} \cup X_{02} \cdot X'_{20} \\
Y_{01} & = & X_{00} \cdot X'_{01} \cup X_{01} \cdot X'_{00} \cup X_{01} \cdot X'_{11} \cup X_{02} \cdot X'_{10} \\
Y_{10} & = & X_{00} \cdot X'_{10} \cup X_{10} \cdot X'_{00} \cup X_{11} \cdot X'_{10} \cup X_{01} \cdot X'_{20} \\
Y_{11} & = & X_{10} \cdot X'_{01} \cup X_{00} \cdot X'_{11} \cup X_{11} \cdot X'_{00} \cup X_{11} \cdot X'_{11} \\
 & & \cup X_{00} \cdot X'_{00} \cup X_{01} \cdot X'_{10} \cup X_{02} \cdot X'_{20} \\
Y_{02} & = & X_{01} \cdot X'_{01} \cup X_{00} \cdot X'_{02} \cup X_{02} \cdot X'_{00} \cup X_{02} \cdot X'_{11} \\
Y_{20} & = & X_{10} \cdot X'_{10} \cup X_{00} \cdot X'_{20} \cup X_{20} \cdot X'_{00} \cup X_{11} \cdot X'_{20}.
\end{array}$$

The following theorem states that the composition of two subTRSs of T_1 can be expressed effectively by a subTRS of T_2, and the composition of two subTRSs of T_2 can be approximated by a subTRS of T_2.

▶ **Theorem 4.**

1. *If $R, S \subseteq T_1$ then $\to_{\mathsf{comp}(R,S)} = \to_R \cdot \to_S$.*
2. *If $R, S \subseteq T_2$ then $\to_{\mathsf{comp}(R,S)} \subseteq \to_R \cdot \to_S$.*

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 5–15

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Proof. We start by proving part (2). Let $R = R(X_{00}, X_{01}, X_{10}, X_{11}, X_{02}, X_{20})$ and $S = R(X'_{00}, X'_{01}, X'_{10}, X'_{11}, X'_{02}, X'_{20})$. Assume $t \to_{\mathsf{comp}(R,S)} u$, then there t rewrites to u with respect to a rule $f^p(a) \to f^q(b)$ for $(a,b) \in Y_{pq}$, for $(p,q) \in \{(0,0),(0,1),(1,0),(1,1),(0,2),(2,0)\}$ and Y_{pq} from the definition of $\mathsf{comp}(R,S)$. Observe that for every part $Y_{pq} = \cdots \cup X_{ij} \cdot X'_{km} \cup \cdots$ in this definition of Y_{pq}, the values p and q satisfy either $p = \max(i, i+k-j)$ and $q = \max(m, m+j-k)$ or $p = \max(i, i+k-j)+1$ and $q = \max(m, m+j-k)+1$. Assume $t \to_{R(Y_{00},Y_{01},Y_{10},Y_{11},Y_{02},Y_{20})} u$. Then $t \to u$ with respect to a rule from $R_{pqY_{pq}}$ for some $(p,q) \in \{(0,0),(0,1),(1,0),(1,1),(0,2),(2,0)\}$, hence from a rule from $R_{pqX_{ij}\cdot X'_{km}}$ for some i,j,k,m satisfying either $p = \max(i, i+k-j)$ and $q = \max(m, m+j-k)$ or $p = \max(i, i+k-j)+1$ and $q = \max(m, m+j-k)+1$. For the first case according to Theorem 2 we conclude $t \to_{R_{ijX_{ij}}} \cdot \to_{R_{kmX'_{km}}} u$ for some $(i,j),(k,m) \in \{(0,0),(0,1),(1,0),(1,1),(0,2),(2,0)\}$. For the second case by Lemma 3 we conclude $t \to u$ with respect to a rule from $R_{p-1,q-1,Y_{pq}}$, hence again by Theorem 2 we conclude $t \to_{R_{ijX_{ij}}} \cdot \to_{R_{kmX'_{km}}} u$. Hence

$$t \to_{R(X_{00},X_{01},X_{10},X_{11},X_{02},X_{20})} \cdot \to_{R(X'_{00},X'_{01},X'_{10},X'_{11},X'_{02},X'_{20})} u,$$

concluding the proof of part (2).

For the proof of part (1) we only need to prove $\to_R \cdot \to_S \subseteq \to_{\mathsf{comp}(R,S)}$ as the other direction already follows from part (2). Assume $t \to_R \cdot \to_S u$ for $R = R(X_{00}, X_{01}, X_{10}) \in T_1$ and $S = R(X'_{00}, X'_{01}, X'_{10}) \in T_1$. Then there is a rule $f^i(a) \to f^j(b)$ in R for which $(a,b) \in X_{ij}$ by which t rewrites to a term v and a rule $f^k(b) \to f^m(c)$ for which $(b,c) \in X'_{km}$ by which v rewrites to u. Here $(i,j),(k,m) \in \{(0,0),(0,1),(1,0)\}$. By Theorem 2 we conclude that $t \to_{R_{pq\{a,c\}}} u$ for $p = \max(i, i+k-j)$ and $q = \max(m, m+j-k)$. One checks that in all cases $(p,q) \in \{(0,0),(0,1),(1,0),(1,1),(0,2),(2,0)\}$, and $X_{ij} \cdot X'_{km}$ occurs in the definition of Y_{pq}. Hence $t \to_{\mathsf{comp}(R,S)} u$, concluding the proof. ◀

4 Local confluence

Composition of two subTRSs of T_2 is typically not expressed by a subTRS of T_2. For instance, composing $a \to f(f(b))$ with $b \to f(c)$ yields $a \to f(f(f(c)))$, being outside the scope of T_2. So when restricting to subTRSs of T_2, for composition we can only approximate the result by a subTRS of T_2 as we expressed in part (2) of Theorem 4. Fortunately, this is sufficient for expressing a slight strengthening of the important notion of local confluence, as is expressed in the following theorem. It makes use of the inverse inv and the reflexive closure rc of a ground TRS over $\Sigma = \{f\} \cup A$ defined as follows:

$$\mathsf{inv}(R) = \{r \to \ell \mid \ell \to r \in R\}, \quad \text{and}$$

$$\mathsf{rc}(R) = R \cup \{a \to a \mid a \in A\}.$$

For these operations we have the following obvious lemma.

▶ **Lemma 5.** *If $R \subseteq T_1$ then $\mathsf{inv}(R) \subseteq T_1$ and $\mathsf{rc}(R) \subseteq T_1$.*
If $R \subseteq T_2$ then $\mathsf{inv}(R) \subseteq T_2$ and $\mathsf{rc}(R) \subseteq T_2$.

▶ **Theorem 6.** *Let $R \subseteq T_1$, $R_1 = \mathsf{rc}(R)$, $R_{i+1} = \mathsf{comp}(R_i, R_i)$ for $i > 0$, and*

$$\mathsf{comp}(\mathsf{inv}(R), R) \subseteq \mathsf{comp}(R_i, \mathsf{inv}(R_i))$$

for some $i > 0$. Then R is locally confluent.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 6–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Proof. By Lemma 5 we have $R_1 = \mathsf{rc}(R) \subseteq T_1 \subseteq T_2$; using Theorem 4, part (2), one easily proves by induction that $R_i \subseteq T_2$ and $\to_{R_i} \subseteq (\to_{\mathsf{rc}(R)})^{2^{i-1}} \subseteq \to_R^*$, for all $i > 0$.

We have to prove that $\leftarrow_R \cdot \to_R \subseteq \to_R^* \cdot \leftarrow_R^*$. By Lemma 5 we have $\mathsf{inv}(R) \subseteq T_1$. As $\leftarrow_R = \to_{\mathsf{inv}(R)}$, by Theorem 4, part (1), we have $\leftarrow_R \cdot \to_R = \to_{\mathsf{comp}(\mathsf{inv}(R),R)}$. As $R_i, \mathsf{inv}(R_i) \subseteq T_2$, by Theorem 4, part (2), we conclude $\to_{\mathsf{comp}(R_i,\mathsf{inv}(R_i))} \subseteq \to_{R_i} \cdot \leftarrow_{R_i}$. Combining all these yields

$$\leftarrow_R \cdot \to_R = \to_{\mathsf{comp}(\mathsf{inv}(R),R)} \subseteq \to_{\mathsf{comp}(R_i,\mathsf{inv}(R_i))} \subseteq \to_{R_i} \cdot \leftarrow_{R_i} \subseteq \to_R^* \cdot \leftarrow_R^*,$$

concluding the proof. ◀

We want to stress that for Theorem 6 the requirement $R \subseteq T_1$ is essential: if this requirement is relaxed to $R \subseteq T_2$ the theorem does not hold. For instance let $R = \{a \to f(f(b)), a \to f(f(c))\}$. Then R is not locally confluent since $f(f(b)) \leftarrow_R a \to_R f(f(c))$ and $f(f(b))$ and $f(f(c))$ are distinct normal forms. However, $\mathsf{comp}(\mathsf{inv}(R), R) = \emptyset$ since in this instance of the definition of comp only X_{20} and X'_{02} are non-empty and all other X_{ij}, X'_{ij} are empty, by which all Y_{ij} are empty. So the condition $\mathsf{comp}(\mathsf{inv}(R), R) \subseteq \mathsf{comp}(R_i, \mathsf{inv}(R_i))$ of the theorem holds.

Next, we observe that the local confluence criterion of Theorem 6 is sufficient, but not necessary. As an example consider $R = \{a \to f(b), f(a) \to c, b \to f(c), c \to f(c)\} \subseteq T_1$. This TRS is locally confluent as the only critical pair $[c, f(f(b))]$ converges via $c \to^3 f^3(c) \leftarrow f^2(b)$. However, $c \to f(f(b)) \in \mathsf{comp}(\mathsf{inv}(R), R)$, but $c \to f(f(b))$ is not in $\mathsf{comp}(R_i, \mathsf{inv}(R_i))$ for any i, since the convergence of the critical pair requires $c \to_{R_i} f^3(c)$, which is impossible for $R_i \subseteq T_2$.

Finally, we observe that a criterion for confluence similar to that of Theorem 6 for local confluence does not hold. More precisely, from

$$\mathsf{comp}(\mathsf{inv}(R_i), R_i) \subseteq \mathsf{comp}(R_i, \mathsf{inv}(R_i))$$

we can not conclude confluence. Consider the TRS R consisting of the three rules $1 \to f(1)$, $1 \to f(2)$, $f(1) \to 1$, then $\mathsf{comp}(\mathsf{inv}(R_2), R_2) \subseteq \mathsf{comp}(R_2, \mathsf{inv}(R_2))$ holds, while R is not confluent since 1 had two distinct normal forms $f(2)$ and $f(f(2))$.

After these negative remarks we conclude this section positive: with the same proof Theorem 6 generalizes to proving any requirement of the shape $A \subseteq B$ for A and B composed from comp, inv and rc and any number of basic TRSs, as long as in this composition A contains comp at most once.

5 Termination

In the last section we saw that when restricting to T_2, composition can only be approximated as the result may be outside T_2. As a consequence, we failed to give a full characterization of local confluence and only could give a sufficient but not necessary criterion. In this section we show that by using functions from ground terms to real numbers, we can give a full characterization of termination of any finite ground TRS over $\Sigma = \{f\} \cup A$. This characterization is expressed as the existence of a number of real values satisfying a number of linear inequalities, so it is a feasibility requirement in linear programming. Where in the earlier sections all requirements were in propositional satisfiability (SAT), here we obtained a formula in a richer theory: SMT (satisfiability modulo theories), for the theory consisting of linear inequalities.

▶ **Theorem 7.** *A ground TRS R over $\{f\} \cup A$ for A finite is terminating if and only if a map $W : A \to \mathbf{R}$ exists such that $W(a) + n > W(b) + k$ for every $f^n(a) \to f^k(b) \in R$.*

Proof. For the 'if'-part assume that such a map W exists. Assume R admits an infinite reduction $t_0 \to t_1 \to t_2 \to \cdots$. For a ground term $t = f^k(a)$ write $W(t) = W(a) + k$. So from the given property we conclude $W(t_i) > W(t_j)$ for $i > j$. Since A is finite there exists $a \in A$ such that infinitely many t_i's are of the shape $f^k(a)$. Among these infinitely many the corresponding k cannot go down forever, so there exists $i < j$ such that $t_i = f^{k(i)}(a)$ and $t_j = f^{k(j)}(a)$ with $k(i) \leq k(j)$. So $W(t_i) \leq W(t_j)$, contradicting our observation $W(t_i) > W(t_j)$.

Conversely assume R is terminating; we have to find a corresponding map W. We do this by proving the following claim by induction on $\#A'$.

> **Claim:** For $A' \subseteq A$ there exists a map $W : A' \to \mathbf{R}$ such that if $f^n(a) \to_R^+ f^k(b)$ for $a, b \in A'$, then $W(a) + n > W(b) + k$.

For $\#A' = 0$ this is vacuously true. For the induction step assume that $A' = A'' \cup \{a'\}$; as the induction hypothesis we assume that the claim holds for A'' and we have to prove the claim for A'. Define

$$x = \min\{W(a) + n - k \mid f^n(a) \to_R^+ f^k(a'), a \in A'', k, n \geq 0\}, \text{and}$$

$$y = \max\{W(b) + k - n \mid f^n(a') \to_R^+ f^k(b), b \in A'', k, n \geq 0\}.$$

So there exist reductions $f^n(a) \to_R^+ f^k(a')$ and $f^{n'}(a') \to_R^+ f^{k'}(b)$ satisfying $x = W(a)+n-k$ and $y = W(b) + k' - n'$. Combining these reductions we obtain $f^{n+n'}(a) \to_R^+ f^{k+n'}(a') \to_R^+ f^{k+k'}(b)$. Since $a, b \in A''$ the induction hypothesis yields $x + n' + k = W(a) + n + n' > W(b) + k + k' = y + n' + k$, hence $x > y$. Here $W : A'' \to \mathbf{R}$. We extend the domain of W to A' by defining $W(a') = (x + y)/2$. It remains to prove that if $f^n(a) \to_R^+ f^k(b)$ for $a, b \in A'$, then $W(a) + n > W(b) + k$, which we do by case analysis.

If $a, b \in A''$ this follows from the induction hypothesis.

If $a = b = a'$ then from $f^n(a') \to_R^+ f^k(a')$ and termination of R we conclude that $n > k$, hence $W(a) + n > W(b) + k$.

If $a \in A''$ and $b = a'$ then from $f^n(a) \to_R^+ f^k(a')$ and the definition of x we conclude $W(a) + n - k \geq x > (x + y)/2 = W(a') = W(b)$, so $W(a) + n > W(b) + k$.

Finally, if $a = a'$ and $b \in A''$ then from $f^n(a') \to_R^+ f^k(b)$ and the definition of y we conclude $W(b) + k - n \leq y < (x + y)/2 = W(a') = W(a)$, so $W(a) + n > W(b) + k$.

◀

6 Projection to finite ARSs

In this section we show how a TRS can be projected to a finite ARS in such a way that some properties preserve. In particular, the projection of a confluent TRS is again confluent, so if the projection is not confluent, we can conclude that neither the original TRS is confluent.

Fix an integer $m > 1$. For a signature $\Sigma = \{f\} \cup A$, where f has arity 1 and all elements of the finite set A have arity zero, we define a new signature $\Sigma' = \{a_i \mid a \in A \wedge 0 \leq i < m\}$, in which all elements of Σ' are constants. For a ground TRS R over Σ we define the finite ARS R' on Σ' defined by $a_i \to_{R'} b_j$ if and only if there exists a rule $f^n(a) \to f^k(b)$ in R for which $i - j \equiv n - k \mod m$.

We define $\pi : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$ by defining $\pi(f^n(a)) = a_i$ for i being the unique value $0 \leq i < m$ satisfying $i \equiv n \mod m$, for all $a \in A$, $n \geq 0$.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 8–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

▶ **Lemma 8.** *In the above setting, if* $t \to_R u$ *for* $t, u \in \mathcal{T}(\Sigma)$*, then* $\pi(t) \to_{R'} \pi(u)$*.*

Conversely, if $t' \to_{R'} u'$ *for* $t', u' \in \mathcal{T}(\Sigma')$*, then there exist* $t, u \in \mathcal{T}(\Sigma)$ *such that* $t' = \pi(t)$*,* $u' = \pi(u)$ *and* $t \to_R u$*.*

Proof. Since $t \to_R u$, we have $t = f^{n+i}(a)$ and $u = f^{k+i}(b)$ for some $i \geq 0$ and some rule $f^n(a) \to f^k(b)$ in R. Since $(n+i) - (k+i) \equiv n - k \mod m$, we conclude that $\pi(t) \to \pi(u)$ is a rule in R', so $\pi(t) \to_{R'} \pi(u)$.

For proving the second claim let $a_i = t' \to_{R'} u' = b_j$ for $a, b \in A, 0 \leq i, j < m$. Then by definition there exists a rule $f^n(a) \to f^k(b)$ in R for which $i - j \equiv n - k \mod m$. Choose $p \geq 0$ such that $n + p \equiv i \mod m$. Then using $i - j \equiv n - k \mod m$ one concludes $k + p \equiv j \mod m$. Choose $t = f^{n+p}(a)$ and $u = f^{k+p}(b)$, then $t' = a_i = \pi(t)$, $u' = b_j = \pi(u)$ and $t \to_R u$. ◀

▶ **Theorem 9.** *In the above setting, if* R *is confluent, then* R' *is confluent too.*

Proof. Assume R is confluent. Assume $u' \leftarrow^*_{R'} t' \to^*_{R'} v'$ for $t', u', v' \in \mathcal{T}(\Sigma')$; we have to prove that $w' \in \mathcal{T}(\Sigma')$ exists such that $u' \to^*_{R'} w' \leftarrow^*_{R'} v'$. Applying the second part of Lemma 8 to every step in $u' \leftarrow^*_{R'} t' \to^*_{R'} v'$ yields $t, u, v \in \mathcal{T}(\Sigma)$ such that $u \leftarrow^*_R t \to^*_R v$ and $u' = \pi(u)$ and $v' = \pi(v)$. Applying confluence of R yields $w \in \mathcal{T}(\Sigma)$ such that $u \to^*_R w \leftarrow^*_R v$. Now choose $w' = \pi(w)$; applying the first part of Lemma 8 to every step of $u \to^*_R w \leftarrow^*_R v$ yields $u' = \pi(u) \to^*_{R'} w' = \pi(w) \leftarrow^*_{R'} \pi(v) = v'$. ◀

7 Implementation

In earlier work [6] we developed a tool `Carpa` reading a list of desired properties in a corresponding input language, looking for finite ARSs. In this input language one can specify the number of ARSs one is looking for, the number of elements on which these ARSs act, several constructions to combine ARSs like union, composition, peaks and valleys, and all kind of properties like (local) confluence and termination. When `Carpa` is executed on such an input then it tries to construct a corresponding set of finite ARSs satisfying all specified properties, and in case of success it yields these ARSs as output.

Now we developed a prototype tool `Carpa+` that does the same for a simple class of TRSs rather than finite ARSs, namely the class of subTRSs of T_1, exploiting the theory as presented in this paper. As some constructions are specific for TRSs, like the projection to finite ARSs, we decided to keep the input language for `Carpa+` closely related to that for `Carpa`, but not exactly the same.

Just like `Carpa`, also `Carpa+` can be downloaded in zip format from

`http://www.win.tue.nl/~hzantema/carpa.html`

including the source code, a Linux executable, a file `Readme` with basic instructions, and encoding of all examples in this paper. For instance, after extracting the zip file in a directory in a Linux environment, by calling `./carpa+ ex10`, the tool is applied on Example 10 being the leading example of this paper, and the output as announced in the introduction is generated.

The main ingredients are:

- `subs`, where `subs`(R, S) means that $\to_R \subseteq \to_S$ holds,
- `nsubs`, where `nsubs`(R, S) means that $\to_R \subseteq \to_S$ does not hold,
- `sn`, where `sn`(R) means that R is terminating,
- `nrrules`, where `nrrules`(R, k) means that R has no more than k rules,

- `union`, where `union`(R, S) represents the TRS $R \cup S$,
- `comp`, where `comp`(R, S) represents the approximation of composition of R and S as presented in Section 3,
- `inv`, where `inv`(R) represents the inverse of R,
- `peak`, where `peak`(R, S) represents `comp`$($`inv`$(R), S)$,
- `val`, where `val`(R, S) represents `comp`$(R,$`inv`$, S)$,
- `rc`, where `rc`(R) represents the reflexive closure $R \cup I$ of R, and
- `mod2`, where `mod2`(R) represents the projection of R to a finite ARS as presented in Section 6, for $m = 2$.

The names `x1`, `x2`, `x3`,... are used as variable names. For instance,

```
x1=peak(1,1)
x2=rc(1)
x2=comp(x2,x2)
x2=comp(x2,x2)
x2=val(x2,x2)
subs(x1,x2)
```

is a correct specification for local confluence of TRS number 1, according to Theorem 6, for $i = 3$, and in case one line `x2=comp(x2,x2)` is removed it is still correct according to the same theorem for $i = 2$.

The full specification starts by two numbers: first the number of constants to be used in the TRS to be found, being the number $n = \#A$ in this paper. Second, it reads the number of basic TRSs to be looked for. Our tool `Carpa` reads a list of requirements in this format, and builds a formula for it representing these requirements in the way as described in this paper. More precisely, for every basic TRS $3n^2$ boolean variables are introduced: for every pair $(a, b) \in A \times A$ three variables expressing whether the rules $a \to b$, $a \to f(b)$ and $f(a) \to b$ are in the TRS. When dealing with composition, auxiliary subTRSs of T_2 are introduced, each represented by $6n^2$ boolean variables, indicating whether the rules $a \to b$, $a \to f(b)$, $f(a) \to b$, $f(a) \to f(b)$, $a \to f(f(b))$ and $f(f(a)) \to b$ are in the TRS for every pair $(a, b) \in A \times A$. According to Theorem 4, part 1, every composition of two subTRSs of T_1 can be expressed exactly as such a subTRS of T_2. Also subTRSs of T_2 can be composed, but as observed, typically the result will not be expressible by a subTRS of T_2 any more. As in our implementation we restrict to expressing subTRSs of T_2, for composing two subTRSs of T_2 the result is not expressed exactly, but only approximated as indicated by Theorem 4, part 2. In all cases, the requirements that the TRS represented by the new variable is equal to `comp` applied on the arguments, are expressed in proposition logic and are added to the formula.

For every assignment a new TRS is created; either being a subTRS of T_1 represented by $3n^2$ boolean variables (type 1), or a subTRS of T_2 represented by $6n^2$ boolean variables (type 2). This type is statically detected from the input: the basic TRSs are of type 1; as long as only `inv`, `rc` or `union` is applied the type remains 1, but as soon as soon as `comp`, `peak` or `val` comes in the type is 2, and the type of the union of two TRSs is the maximum of the argument types.

For every call of `sn` n real valued variables are introduced, and the conditional inequalities indicated by Theorem 7 are added to the formula. Here the condition of the inequality $W(a) + n > W(b) + k$ is the existence of the rule $f^n(a) \to f^k(b)$ in the TRS, corresponding to the boolean variable representing this existence.

Finally, by applying `mod2` the new variable is not of type 1 or type 2 any more, but switches to an ARS on $2n$ elements, represented by $4n^2$ boolean variables. The equality

between this ARS and the ARS R' as defined in Section 6 is expressed in propositional logic and added to the formula. In forthcoming lines the full `Carpa` syntax can be used to express requirements on this ARS, and a copy of `Carpa` is called to generate these requirements in propositional logic to be added to the formula. For instance, in `Carpa` one may call `ncr` to express that it argument is not confluent.

After building this formula, `Carpa+` calls the SMT solver `yices`. In case `yices` observes that the formula is unsatisfiable, `Carpa` reports that the list of requirements has no solution. In case the formula turns out to be satisfiable, `Carpa+` inspects the resulting satisfying assignment and reports the resulting TRSs as indicated by the values of the variables representing the basic TRSs.

▶ **Example 10.** As announced in the introduction, let's look for a TRS that is locally confluent but not confluent, and for which the inverse is terminating. Choose the number of constants to be three, then the full input for this problem reads:

```
3 % nr of constants
1 % nr of TRSs
x1=inv(1)
sn(x1)
x1=peak(1,1)
x2=rc(1)
x2=comp(x2,x2)
x2=comp(x2,x2)
x2=val(x2,x2)
subs(x1,x2)
x1=mod2(1)
ncr(x1)
```

and `Carpa+` generates the following output

```
1 -> 2
1 -> f(1)
2 -> f(3)
3 -> f(2)
```

indeed being the TRS for which in the introduction validity of the properties was explained and the reduction graph was presented.

This example has four rules. If one wonders whether it can be done with less rules, one may add the rule `nrrules(1,3)` to the input, and then `Carpa+` reports `no solution`. This shows that no smaller example exists satisfying the requirements, being slightly stronger than the original requirements: both local confluence and non-confluence were approximated by stronger properties. A similar remark holds for finding an example with less constants: if the '3' in the first input line is replaced by '2', then `Carpa+` reports `no solution`, showing that it can not be done with less than three constants.

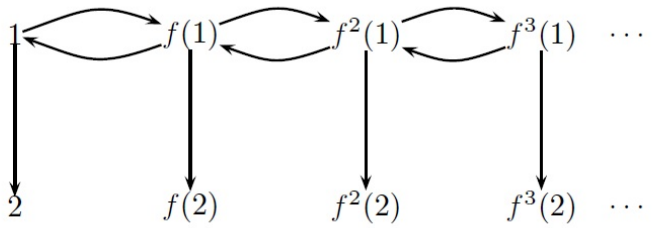
8 More examples

▶ **Example 11.** The ARS example in the introduction of a system that is locally confluent but not confluent contains four constants and four rules. Now we wonder whether it can be done in less when using TRS rather than ARS. We simply copy the specification of Example 10, and remove the requirement of termination of the inverse, and apply `Carpa+`

on variations in the number of constants and the number of rules. The smallest solution found has only two constants and three rules, and reads

```
1 -> 2
1 -> f(1)
f(1) -> 1
```

Indeed, this system is easily checked to be locally confluent, while it is not confluent as 1 has infinitely many normal forms. The system has the following reduction graph:



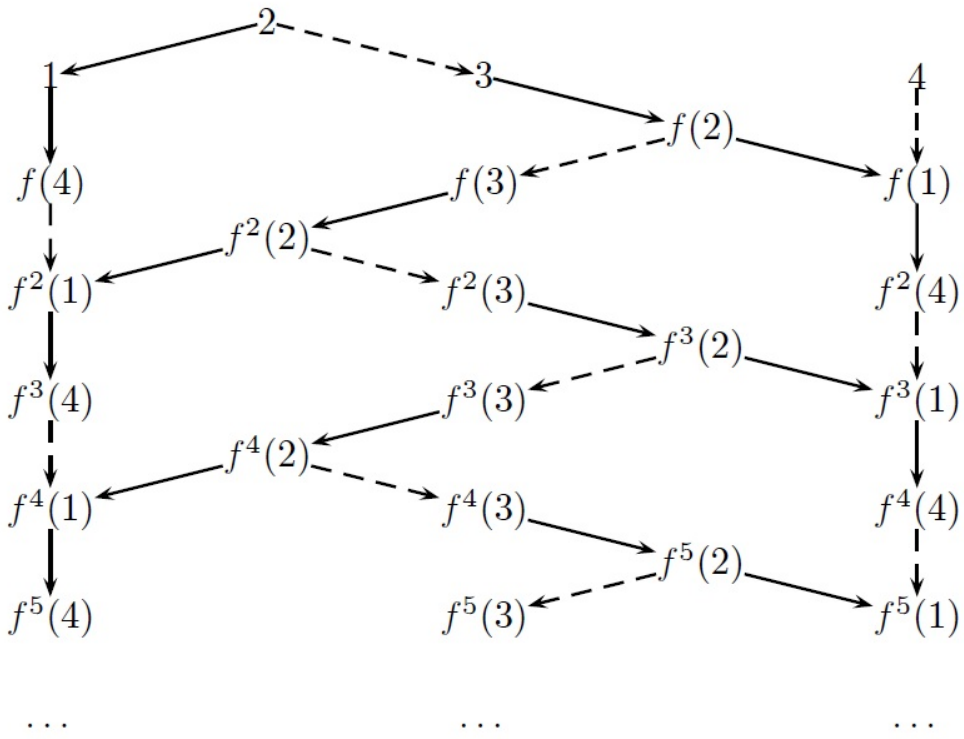
▶ **Example 12.** In Example 10 we have the rule $1 \to f(1)$, by which it is impossible to split up the TRS in two terminating subTRSs. Now we keep the same requirements for TRS 1 (locally confluent, not confluent, and the inverse is terminating), but add some more requirements. We are looking for three TRSs rather than one, and require that TRS 1 is the union of terminating TRSs 2 and 3, specified by

```
subs(2,1)
subs(3,1)
x1=union(2,3)
subs(1,x1)
sn(2)
sn(3)
```

Moreover, we add the requirements that the TRSs 2 and 3 both are locally confluent, specified in the same way as for TRS 1. The full specification for specifying three constants gives no solution, but if the number of constants is replaced by four, then `Carpa+` generates the following output

```
TRS 1:
2 -> 1
2 -> 3
1 -> f(4)
3 -> f(2)
4 -> f(1)
TRS 2:
2 -> 3
4 -> f(1)
TRS 3:
2 -> 1
1 -> f(4)
3 -> f(2)
```

indeed giving TRSs satisfying all required properties. The following picture shows the reduction graph, in which reduction steps of TRS 2 are drawn dashed and reduction steps of TRS 3 are drawn solid.



▶ **Example 13.** We wonder whether there are terminating TRSs R and S such that $\to_R \cdot \to_S \subseteq \to_S \cdot \to_R^+$, but not $\to_R \cdot \to_S \subseteq \to_S \cdot \to_R$. For this last requirement we may use `nsubs` since by using `comp` we have exact descriptions for both $\to_R \cdots \to_S$ and $\to_S \cdot \to_R$ as subTRSs of T_2. Also by expressing $\to_R \cdot \to_S \subseteq \to_S \cdot \to_R^+$ by `comp` we have a sound description as was remarked at the very end of Section 4. So we may apply `Carpa+` on

```
2 % nr of constants
2 % nr of TRSs
sn(1)
sn(2)
x1=comp(1,2)
x2=comp(2,1)
nsubs(x1,x2)
x3=rc(1)
x3=comp(x3,x3)
x2=comp(x2,x3)
subs(x1,x2)
```

indeed yielding a solution. After playing around with `nrrules` the following smallest solution is found:

```
TRS 1:
1 -> 2
f(1) -> 1
TRS 2:
f(1) -> 1
f(1) -> 2
```

indeed satisfying the required properties, indicated by

$$f(f(1)) \to_1 f(1) \to_2 2 \quad \text{and} \quad f(f(1)) \to_2 f(1) \to_1 1 \to_1 2.$$

9 Conclusion

In earlier work [6] we described how to find finite ARSs fully automatically satisfying a given list of properties, based upon transforming the requirements to a SAT problem. This work was inspired and initiated by [3, 4]. In this paper we did some first steps to extend this work to automatically finding TRSs rather than finite ARSs. This turned out to be a hard job: many questions become much harder when lifting a setting of finitely many elements to a setting of infinitely many terms. We decided to restrict to a very simple class of term rewriting systems: we only allow rewrite rules of the shape $a \to b$, $a \to f(b)$ or $f(a) \to b$, for a single unary symbol f and a finite set of constants. On the one hand this is a strong restriction, on the other hand this class allows terminating TRSs for which the inverse is non-terminating, representing a typical property of TRSs that does not occur for finite ARSs. Our goal was to automatically find a TRS that is locally confluent but not confluent, and for which the inverse is terminating. For doing so we had to describe the chosen class of TRSs by a series of boolean variables, and have to describe or approximate the properties by propositional satisfiability or a suitable extension. For local confluence we found an approximation based on approximating composition of rewrite steps; for non-confluence we found an approximation based on projection to a finite ARS. For termination we found an exact description by a propositional formula in linear inequalities, hence extending the underlying SAT framework to SMT. By applying the SMT solver `yices` to the resulting formula indeed indeed our goal was reached. We developed a prototype tool `Carpa+` in which the requirements are entered, a corresponding formula is composed, `yices` is called on this formula, and the output is inspected to obtain the desired result. This approach also applies on some variants and other properties, but we fully realize that as a first step this is more a case study than a general tool for finding TRSs with a given list of properties. Next steps in this direction would include both extension to richer classes of TRSs and better approximations of the basic rewriting properties.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 14–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

References

1 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

2 B. Felgenhauer. Deciding Confluence of Ground Term Rewrite Systems in Cubic Time. In Ashish Tiwari, editor, *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, volume 15 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 165–175, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

3 A. Stump, G. Kimmell, and R. El Haj Omar. Type Preservation as a Confluence Problem. In Manfred Schmidt-Schauß, editor, *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA)*, volume 10 of *LIPIcs*, pages 345–360, 2011.

4 A. Stump, G. Kimmell, H. Zantema, and R. El Haj Omar. A rewriting view of simple typing. *Logical Methods in Computer Science*, 2013. Accepted.

5 Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 2003.

6 H. Zantema. Finding small counter examples for abstract rewriting properties. *Mathematical Structures in Computer Science*, 2013. Accepted, preliminary version available via `http://www.win.tue.nl/~hzantema/carpa.html`.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 15–15

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany