Guillem Godoy
Florent Jacquemard

Unique Normalization for Shallow TRS

Research Report LSV-08-21

June 2008

Laboratoire Spécification et Vérification

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Unique Normalization for Shallow TRS

Guillem Godoy[1] and Florent Jacquemard[2]

[1] Technical University of Catalonia, Jordi Girona 1, Barcelona, Spain
ggodoy@lsi.upc.edu
[2] INRIA Saclay & LSV, UMR CNRS/ENS Cachan, France
florent.jacquemard@lsv.ens-cachan.fr

Abstract. Computation with a term rewrite system (TRS) consists in the application of its rules from a given starting term until a normal form is reached, which is considered the result of the computation. The unique normalization (UN) property for a TRS R states that any starting term can reach at most one normal form when R is used, i.e. that the computation with R is unique.
We study the decidability of this property for classes of TRS defined by syntactic restrictions such as linearity (variables can occur only once in each side of the rules), flatness (sides of the rules have depth at most one) and shallowness (variables occur at depth at most one in the rules). We prove that UN is decidable in polynomial time for shallow and linear TRS, using tree automata techniques. This result is very near to the limits of decidability, since this property is known undecidable even for very restricted classes like right-ground TRS, flat TRS and also right-flat and linear TRS. We also show that that UN is even undecidable for flat and right-linear TRS. The latter result is in contrast with the fact that many other natural properties like reachability, termination, confluence, weak normalization... are decidable for this class of TRS.

Introduction

Term rewriting is a Turing-complete model of computation. A term rewrite system (TRS) is a set of oriented equations on terms $s \to t$ called rules. A rule can be applied to a term by replacing a subterm, which is an instantiation of a left-hand side of the rule, by the corresponding instantiation of the right-hand side. Computation with a TRS consists in the application of its rules from a given starting term until a normal form is reached, i.e. a term that cannot be rewritten any more, which is usually considered as the result of the computation.

The unique normalization (UN) property for a TRS $\mathcal{R}$ states that any starting term can reach at most one normal form when $\mathcal{R}$ is used, i.e. that the computation with $\mathcal{R}$ is unique. Other interesting and very studied properties of TRS are reachability (whether a given term can be derived with a given TRS from another given term), joinability (whether two given terms can be rewritten into one common term), termination (whether there are no infinite derivations from any starting term), confluence (whether any two terms derived from a common one can also be rewritten into another common term), weak normalization (whether any term can be rewritten into a normal form), etc.

In the recent years there have been a big progress on determining decidability of these fundamental properties for several classes of TRS, which are defined by imposing certain syntactic restrictions on the rules. Some of the restrictions usually taken into consideration are groundness (no variable appears in the rules), linearity (variables can occur only once in each side of the rules), flatness (sides of the rules have depth at most one) and shallowness (variables occur at depth at most one in the rules). When these restrictions refer only to one of the sides of the rules, then we talk about left-linearity, right-linearity, left-flatness, etc.

Some of the strongest known results are the following. Reachability and joinability are decidable for right-shallow right-linear TRS [12], and even for weaker restrictions [15]. Termination is decidable for right-shallow right-linear TRS [5] and other variants of syntactic restrictions based on the form of the dependency pairs obtained from a TRS [18]. Confluence is decidable for shallow right-linear TRS [8], and for right-(ground or variable) TRS [7]. The weak normalization problem is decidable for left-shallow left-linear TRS [12], right-shallow linear TRS and shallow right-linear TRS [6].

On the negative side, all of these properties have been proved undecidable for flat TRS [10, 11, 5, 4].

The case of the UN property seems to be more difficult. In [17], a polynomial time algorithm is given for UN and TRS with ground rules. On the negative side, UN is proved undecidable for right-flat linear TRS [6], for right-ground TRS [16] and for flat TRS [4]. In [6] its decidability is left open for flat right-linear TRS.

In this paper, we provide a polynomial time algorithm for deciding UN for shallow and linear TRS (Sections 2 and 3). We also prove (in Section 4) undecidability of UN for flat and right-linear TRS. Our approach for decidability in polynomial time consists in giving a certain characterization of UN. We essentially show that UN is equivalent to the fact that certain regular sets of terms can reach at most one normal form. This characterization can then be checked using tree automata techniques. The proof of undecidability is an adequate adaptation of the reductions appearing in [5, 4].

1 Preliminaries

For basic notions of rewriting see e.g. [1].

Terms Algebra. A *signature* Σ is a finite set of function symbols with arity. We write Σ_m the subset of function symbols of Σ of arity m. Given an infinite set $\mathcal{V}$ of variables, the set of terms built over Σ and $\mathcal{V}$ is denoted $\mathcal{T}(\Sigma, \mathcal{V})$, and the subset of ground terms is denoted $\mathcal{T}(\Sigma)$. The set of variables occurring in a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is denoted $vars(t)$. A *substitution* σ is a mapping from $\mathcal{V}$ to $\mathcal{T}(\Sigma, \mathcal{V})$; its application to a term t is written $t\sigma$, and is the homomorphic extension of σ to $\mathcal{T}(\Sigma, \mathcal{V})$. A *variable renaming* is a substitution from variables to variables.

A term t is identified as usual to a function from its set of *positions* (finite sequences of positive integers) $\mathcal{P}os(t)$ to symbols of Σ and $\mathcal{V}$. We note Λ the

empty string (root position). The length of a position p is denoted $|p|$. The *height* of a term t, denoted $h(t)$, is the maximum of $\{|p| \mid p \in \mathcal{P}os(t)\}$. A subterm of t at position p is written $t|_p$, and the replacement in t of the subterm at position p by u denoted $t[u]_p$. The concatenation of two positions p and p' is denoted pp'. The prefix ordering on position is denoted by $p \leq pp'$ for all p, p'. Two positions p and p' are called *disjoint*, denoted by $p \parallel p'$, if they are incomparable wrt $\leq$.

Term Rewriting. A *term rewriting system* (TRS) $\mathcal{R}$ over a signature Σ is a finite set of rewrite rules $\ell \to r$, where $\ell, r \in \mathcal{T}(\Sigma, \mathcal{V}) - \ell$ and r are respectively called left-hand-side (*lhs*) and right-hand-side (*rhs*) of the rule.

Note that we allow variables in the rhs which do not occur in the corresponding lhs. Such a definition is not standard; however, this technical convenience will allow us to define the inverse of a TRS $\mathcal{R}$ as a TRS $\mathcal{R}^{-1} := \{r \to \ell \mid \ell \to r \in \mathcal{R}\}$.

A term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ rewrites to t by a TRS $\mathcal{R}$ at a position p of s with a substitution σ, denoted $s \xrightarrow[p,\sigma,\mathcal{R}]{} t$ (p and σ may be omitted in this notation) if there is a rewrite rule $\ell \to r \in \mathcal{R}$ such that $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$. In this case, s is said to be $\mathcal{R}$-*reducible*. The set of irreducible terms, also called $\mathcal{R}$-*normal-forms*, is denoted by $\mathsf{NF}_\mathcal{R}$. The transitive and reflexive closure of the relation $\xrightarrow[\mathcal{R}]{}$ is denoted $\xrightarrow[\mathcal{R}]{*}$. Given $L \subseteq \mathcal{T}(\Sigma)$, we note $\mathcal{R}^*(L) = \{t \mid \exists s \in L, s \xrightarrow[\mathcal{R}]{*} t\}$ and $\mathsf{NF}_\mathcal{R}(L) = \mathcal{R}^*(L) \cap \mathsf{NF}_\mathcal{R}$. Some terms $s_1, \ldots, s_n$ are called *joinable* by $\mathcal{R}$ (resp. they have a *common ancestor* wrt $\mathcal{R}$) if there exists a term t such that $s_i \xrightarrow[\mathcal{R}]{*} t$ (resp. $t \xrightarrow[\mathcal{R}]{*} s_i$) for all i in $\{1, \ldots, n\}$.

A TRS is called *linear* if every variable occurs at most once in each term of the rules. It is called *shallow* if variables occur at depth 0 or 1 in the terms of the rules and *flat* if the terms in the rules have height at most 1. Note that when $\mathcal{R}$ is flat (resp. shallow, linear), then $\mathcal{R}^{-1}$ is also flat (resp. shallow, linear).

Tree Automata. A *tree automaton* (TA) $\mathcal{A}$ on a signature Σ is a tuple $(Q, Q^\mathsf{f}, \Delta)$ where Q is a finite set of nullary state symbols, disjoint from Σ, $Q^\mathsf{f} \subseteq Q$ is the subset of final states and Δ is a set of ground rewrite rules of the form: $f(q_1, \ldots, q_m) \to q$, or $q_1 \to q$ (ε-transition) where $f \in \Sigma_m$, and $q_1, \ldots, q_m, q \in Q$ (q is called the *target* state of the rule).

The language of ground terms *accepted* by a TA $\mathcal{A}$ on Σ in a state q is the set $L(\mathcal{A}, q) := \{t \in \mathcal{T}(\Sigma) \mid t \xrightarrow[\Delta]{*} q\}$. The language of $\mathcal{A}$ is $L(\mathcal{A}) := \bigcup_{q \in Q^\mathsf{f}} L(\mathcal{A}, q)$ and a subset of $\mathcal{T}(\Sigma)$ is called *regular* if it is the language of a TA.

We shall use the following classical properties and problems of TA, see [2] for details. The class of regular tree languages is closed under Boolean operations.

Proposition 1. *Given two TA $\mathcal{A}_1$ and $\mathcal{A}_2$ on the same signature Σ, one can construct three TA recognizing respectively $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, and $\mathcal{T}(\Sigma) \setminus L(\mathcal{A}_1)$, whose sizes are respectively linear, quadratic and exponential in the size of $\mathcal{A}_1$ and $\mathcal{A}_2$.*

We will consider the two following decision problems for TA:

Problem: Emptiness. **Instance:** a TA $\mathcal{A}$. **Question:** do we have $L(\mathcal{A}) = \emptyset$?
Problem: Singleton. **Instance:** a TA $\mathcal{A}$. **Question:** do we have $|L(\mathcal{A})| = 1$?

Proposition 2. *The emptiness problem is decidable in linear time. The singleton problem is decidable in polynomial time.*

2 Decidability of UN for Flat and Linear TRS

In this section we prove that UN is decidable in polynomial time for flat and linear TRS (Theorem 1). From now on, we assume fixed a signature Σ and consider its cardinality and the maximal arity of its function symbols as constants. Hence, these values are not considered as part of the input of the problem of decision of UN and is therefore not taken into account in complexity evaluations. This is a common approach when evaluating complexity for TRS problems.

After proving some technical lemmas about rewrite reduction with flat and linear TRS in Section 2.1, we identify in a first step (Section 2.2) some necessary and sufficient conditions for UN for such TRS. The conditions are expressed as properties of some characteristic sets of terms, or more precisely pairs of sets of terms, called fork of languages, which, intuitively, characterize pairs of terms $\langle t_1, t_2 \rangle$ representing intermediate steps in derivations from a common term u towards two distinct normal forms.

In the second step (Section 2.3), we show that the conditions can be decided by reduction to the above tree automata decision problems. The polynomial time upper bound is achieved through a careful analysis of the size of TAs constructed during the reduction and the complexity of the decision procedures.

2.1 Preliminary Results

In this subsection, we first present some notions and technical lemmas concerning the rewriting sequences with flat and linear TRS which will be useful in the proof of Theorem 1.

Definition 1. *Given a derivation $s \xrightarrow[\mathcal{R}]{*} t$ and a position $p \in \mathcal{P}os(s)$, we define $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ recursively on the length of $s \xrightarrow[\mathcal{R}]{*} t$ as follows:*
– If $s|_p \in \Sigma_0$, then $use(p, s \xrightarrow[\mathcal{R}]{*} t) := s|_p$.
– If $s|_p \notin \Sigma_0$ and $s \xrightarrow[\mathcal{R}]{*} t$ has length 0, then $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ is undefined.
– If $s|_p \notin \Sigma_0$ and $s \xrightarrow[\mathcal{R}]{*} t$ is of the form $s \xrightarrow[p_1,\mathcal{R}]{} s' \xrightarrow[\mathcal{R}]{*} t$ for a position p_1 such that $p_1 \geq p$ or $p_1 \parallel p$ then $use(p, s \xrightarrow[\mathcal{R}]{*} t) := use(p, s' \xrightarrow[\mathcal{R}]{*} t)$.
– If $s|_p \notin \Sigma_0$ and $s \xrightarrow[\mathcal{R}]{*} t$ is of the form $s \xrightarrow[p_1,\ell\to r]{} s' \xrightarrow[\mathcal{R}]{*} t$, where $p = p_1.i.p_2$, and $\ell|_i \in \mathcal{V}$, we consider two cases. If $\ell|_i$ does not occur in r, then $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ is undefined. Otherwise, if $\ell|_i = r|_q$ for some $q \in \mathcal{P}os(r)$, then $use(p, s \xrightarrow[\mathcal{R}]{*} t) := use(p_1.q.p_2, s' \xrightarrow[\mathcal{R}]{*} t)$.

Example 1. Let us consider the following flat and linear TRS
$\mathcal{R}_1 = \{x + 0 \to x, s(0) \to c_1, x + c_1 \to s(x), x + y \to y + x, s(c_1) \to 0\}$, and the derivation: $\rho_1 := 0 + s(0) \xrightarrow[\mathcal{R}_1]{} 0 + c_1 \xrightarrow[\mathcal{R}_1]{} s(0) \xrightarrow[\mathcal{R}_1]{} c_1$, and let ρ_1' be its subderivation starting with $0 + c_1$. We have $use(1, \rho_1) = 0$ and $use(2, \rho_1) = use(2, \rho_1') = c_1$. Let $\rho_2 := s(0) + s(0) \xrightarrow[\mathcal{R}_1]{} c_1 + s(0) \xrightarrow[\mathcal{R}_1]{} c_1 + c_1 \xrightarrow[\mathcal{R}_1]{} s(c_1) \xrightarrow[\mathcal{R}_1]{} 0$. We have $use(1, \rho_2) = use(2, \rho_2) = c_1$. ◇

The following two lemmas are proved straightforwardly, following Definition 1, in Appendix A.

Lemma 1. *Given a derivation $s \xrightarrow[\mathcal{R}]{*} t$, a position $p \in \mathcal{P}os(s)$ and a constant c, if $use(p, s \xrightarrow[\mathcal{R}]{*} t) = c$, then $s[c]_p \xrightarrow[\mathcal{R}]{*} t$ and $s|_p \xrightarrow[\mathcal{R}]{*} c$.*

Lemma 2. *Let $s \xrightarrow[\mathcal{R}]{} t$ be a derivation, p be a position of s such that $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ is undefined, and x be a variable. Then, either $s[x]_p \xrightarrow[\mathcal{R}]{*} t$, or there exists a position $q \in \mathcal{P}os(t)$ such that $s[x]_p \xrightarrow[\mathcal{R}]{*} t[x]_q$ and $s|_p \xrightarrow[\mathcal{R}]{*} t|_q$.*

2.2 Necessary and sufficient conditions

Definition 2. *A* fork *of $\mathcal{R}$ is a pair of terms $\langle f(s_1, \ldots, s_n), f(t_1, \ldots, t_n)\rangle$, with $n \geq 0$, such that for all $i \in \{1, \ldots, n\}$, either s_i and t_i are the same variable of $\mathcal{V}$, or s_i is a constant and $t_i \xrightarrow[\mathcal{R}]{*} s_i$, or t_i is a constant and $s_i \xrightarrow[\mathcal{R}]{*} t_i$.*

Example 2. $\langle x + (0 + s(0)), x + c_1\rangle$, $\langle c_1 + x, s(0) + x\rangle$ and $\langle s(s(0)), 0\rangle$ are forks of the TRS $\mathcal{R}_1$ given in Example 1. ◇

Proposition 3. *$\mathcal{R}$ is UN if and only if for all fork $\langle s, t\rangle$ of $\mathcal{R}$ and all $\mathcal{R}$-normal forms s', t', such that $s \xrightarrow[\mathcal{R}]{*} s'$ and $t \xrightarrow[\mathcal{R}]{*} t'$, it holds that $s' = t'$.*

Proof. First, we show that the condition is necessary for unique normalization proceeding by contradiction. Assume that there exists a fork $\langle s, t\rangle$ and two different normal forms s' and t' such that $s \xrightarrow[\mathcal{R}]{*} s'$ and $t \xrightarrow[\mathcal{R}]{*} t'$. It suffices to construct a term u reaching both s and t in order to prove that $\mathcal{R}$ is not uniquely normalizing. Let s and t be of the form $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$, respectively. We construct $u = f(u_1, \ldots, u_n)$ as follows. For every i in $\{1, \ldots, n\}$, if s_i and t_i are the same variable, then $u_i := s_i$. Otherwise, if $s_i \xrightarrow[\mathcal{R}]{*} t_i$ then $u_i := s_i$. Analogously, if $t_i \xrightarrow[\mathcal{R}]{*} s_i$, then $u_i := t_i$. It is clear from this construction that $u \xrightarrow[\mathcal{R}]{*} s$ and $u \xrightarrow[\mathcal{R}]{*} t$, and this concludes the proof for this direction.

We prove that the condition is sufficient again by contradiction. We assume that $\mathcal{R}$ is not uniquely normalizing and prove that there exists a fork contradicting the statement. We choose a term u minimal in size with two distinct normal forms v and w reachable from it.

The term u cannot be a variable, since variables cannot be rewritten by $\mathcal{R}$. Hence, $u = f(u_1, \ldots, u_n)$ for some $f \in \Sigma_n$ with $n \geq 0$. In order to conclude, it suffices to construct a fork $\langle s, t\rangle$ and two different normal forms s' and t' reachable from s and t, respectively. For the construction of s, t, s' and t' we proceed iteratively as follows, by initializing them, and modifying them along n steps. The invariant is that s' and t' are always different normal forms reachable from s, and t, respectively, and that every $s|_i$ and $t|_i$ are either both the same variable, or the both are $u|_i$, or one of them is a constant reachable from $u|_i$ and the other is $u|_i$, for $i \in \{1, \ldots, n\}$. At the end of the process, $\langle s, t\rangle$ will be a fork.

First, we set $s := u$, $t := u$, $s' := v$ and $t' := w$. After that, for each i in $\{1, \ldots, n\}$, we modify the values of s, t, s' and t' depending on the (un-)definition of $use(i, s \xrightarrow[\mathcal{R}]{*} s')$ and $use(i, t \xrightarrow[\mathcal{R}]{*} t')$.

If $use(i, s \xrightarrow[\mathcal{R}]{*} s')$ is defined to be a constant c, then, by Lemma 1, $s[c]_i \xrightarrow[\mathcal{R}]{*} s'$ and $s|_i \xrightarrow[\mathcal{R}]{*} c$. In this case we just set $s := s[c]_i$ and left s', t and t' unchanged.

The case where $use(i, s \xrightarrow[\mathcal{R}]{*} s')$ is undefined but $use(i, t \xrightarrow[\mathcal{R}]{*} t')$ is defined to a constant is solved analogously to the previous one.

If both $use(i, s \xrightarrow[\mathcal{R}]{*} s')$ and $use(i, t \xrightarrow[\mathcal{R}]{*} t')$ are undefined, let x be a new variable not occurring in s nor t. Then, by Lemma 2, either $s[x]_i \xrightarrow[\mathcal{R}]{*} s'$, or there exists a position q in $\mathcal{P}os(s')$ such that $s[x]_i \xrightarrow[\mathcal{R}]{*} s'[x]_q$ and $s|_i \xrightarrow[\mathcal{R}]{*} s'|_q$. (This is also analogously true for t and t'.) In any case we set $s := s[x]_i$. In the first case we left s' unchanged and in the second case we set $s' := s'[x]_q$. We proceed analogously with t and t'. Note that both $s|_i$ and $t|_i$ are now the variable x, and that the new s' and t' are also normal forms reachable from s and t, respectively. But the preservation of the invariant stating that s' and t' are still different requires an explanation. They could only be equal if the same position q has been replaced to x in both terms, and in such a case, the old values $s'|_q$ and $t'|_q$ must be different. This would imply that the old $s|_i$ and $t|_i$ reach different normal forms, and hence, $u|_i$ can reach two different normal forms. But this is in contradiction with the fact that u is a minimal term in size reaching two different normal forms. □

Checking the hypotheses of Proposition 3 requires to test an infinite number of forks. In order to obtain a decision procedure for UN based on the notion of forks, (and a reduction to tree automata problems), we generalize Definition 2 of forks to sets of terms (Definition 3 below), and generalize Proposition 3 into Proposition 4 accordingly. In the next definition, we write $f(L_1, \ldots, L_n)$, where $f \in \Sigma_n$ and $L_1, \ldots, L_n \subseteq \mathcal{T}(\Sigma, X)$ for the set $\{f(t_1, \ldots, t_n) \mid t_1 \in L_1, \ldots, t_n \in L_n\}$.

Definition 3. *A* fork of languages *wrt a TRS $\mathcal{R}$ is a pair $\langle L, L' \rangle$ of set of terms where L and L' have the form $f(L_1, \ldots, L_n)$ and $f(L'_1, \ldots, L'_n)$, respectively, and for every $i \in \{1, \ldots, n\}$, either $L_i = L'_i = \{x\}$, for some variable x, or $L_i = \{c\}$ and $L'_i = (\mathcal{R}^{-1})^*(\{c\})$, or $L'_i = \{c\}$ and $L_i = (\mathcal{R}^{-1})^*(\{c\})$, for some constant c.*

The following lemma is an immediate consequence of Definition 3 (following the assumption that the maximal arity of a function symbol in Σ is fixed).

Lemma 3. *The number of forks of languages wrt a flat and linear TRS $\mathcal{R}$ is polynomial in the size of $\mathcal{R}$.*

Now, we can state in the next proposition the necessary and sufficient condition that we shall use for the decision of UN.

Proposition 4. *$\mathcal{R}$ is uniquely normalizing if and only if for all fork of languages $\langle L, L' \rangle$ wrt $\mathcal{R}$, if $\mathcal{R}^*(L) \cap \mathsf{NF}_\mathcal{R} \neq \emptyset$ and $\mathcal{R}^*(L') \cap \mathsf{NF}_\mathcal{R} \neq \emptyset$, then $\mathcal{R}^*(L) \cap \mathsf{NF}_\mathcal{R} = \mathcal{R}^*(L') \cap \mathsf{NF}_\mathcal{R} = \{t\}$, for some term t.*

Proof. We prove the left-to-right direction by contradiction. Hence, assume the existence of a fork of languages $\langle L, L' \rangle$ such that there exists $s' \in \mathcal{R}^*(L) \cap \mathsf{NF}_\mathcal{R}$

and $t' \in \mathcal{R}^*(L') \cap \mathsf{NF}_\mathcal{R}$ satisfying $s' \neq t'$. Let $s \in L$ and $t \in L'$ be such that $s \xrightarrow[\mathcal{R}]{*} s'$ and $t \xrightarrow[\mathcal{R}]{*} t'$. By the definitions of fork and fork of languages, it holds that $\langle s, t\rangle$ is a fork. Moreover, s and t reach s' and t', respectively, which are two different normal forms. Hence, by Proposition 3, $\mathcal{R}$ is not uniquely normalizing.

Now, we prove the right-to-left direction again by contradiction. Assume that $\mathcal{R}$ is not uniquely normalizing. Then, by Proposition 3, there exists a fork $\langle s, t\rangle$ and two different normal forms s' and t' reachable from s and t, respectively. Let s and t be of the form $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$, respectively. We define two languages $L = f(L_1, \ldots, L_n)$ and $L' = f(L'_1, \ldots, L'_n)$ as follows. If both s_i and t_i are the same variable x, then we define $L_i = L'_i = \{x\}$. Otherwise, if s_i is a constant and $t_i \xrightarrow[\mathcal{R}]{*} s_i$, then we define $L_i = \{s_i\}$ and $L'_i = (\mathcal{R}^{-1})^*(s_i)$. Analogously, if t_i is a constant and $s_i \xrightarrow[\mathcal{R}]{*} t_i$, then we define $L'_i = \{t_i\}$ and $L'_i = (\mathcal{R}^{-1})^*(t_i)$. With this definition, it is clear that $\langle L, L'\rangle$ is a fork of languages, and that s' and t' are two different normal forms belonging to $\mathcal{R}^*(L)$ and $\mathcal{R}^*(L')$, respectively, concluding the proof of the second direction. □

2.3 Decision of UN

We show now how to decide the conditions of Proposition 4, and thus, how to decide UN for flat and linear TRS, using tree automata techniques.

Lemma 4. *Given a flat and linear TRS $\mathcal{R}$ over Σ, there exists a TA on Σ, of size polynomial in the size of $\mathcal{R}$, recognizing $\mathsf{NF}_\mathcal{R} \cap \mathcal{T}(\Sigma)$.*

Proof. We construct a TA $\mathcal{A}_\mathcal{R} = (Q, Q^\mathsf{f}, \Delta)$ where $Q = Q^\mathsf{f} = \{q_c \mid c \in \Sigma_0 \cap \mathsf{NF}_\mathcal{R}\} \cup \{q\}$, and Δ contains one rule $c \to q_c$ for every $c \in \Sigma_0$, and one rule $f(q_1, \ldots, q_n) \to q$ for all $q_1, \ldots, q_n, q \in Q$ such that the linear term associated to $f(q_1, \ldots, q_n)$ by replacing every occurrence of q_c by c and all occurrences of q by distinct variables is not a left-hand-side of rule of $\mathcal{R}$ (modulo variable renaming). We can show that $L(\mathcal{A}_\mathcal{R}) = \mathsf{NF}_\mathcal{R} \cap \mathcal{T}(\Sigma)$ by induction on terms for both inclusions. □

Note that flatness and linearity are crucial for the above construction in the given complexity bounds. On the one hand, it is known that if $\mathcal{R}$ is not left-linear, then $\mathsf{NF}_\mathcal{R} \cap \mathcal{T}(\Sigma)$ is not a TA language. On the other hand, $\mathsf{NF}_\mathcal{R} \cap \mathcal{T}(\Sigma)$ is a TA language as soon as $\mathcal{R}$ is left-linear, but there is an exponential lower bound on the number of states of a TA recognizing this language [3].

The proof of the following lemma 5 can be found in [9]; it follows a technique called TA completion initiated in [13], where the set of transition rules of $\mathcal{A}$ is completed iteratively, without adding new states.

There exists TA construction for larger classes of TRS than flat and linear, like right-shallow and right-linear TRS [12]. We focus on the flat and linear case here because, as we shall see later, UN is not decidable for these larger classes.

Lemma 5. *Given a TA $\mathcal{A}$ and a flat and linear TRS $\mathcal{R}$, there exists a TA of size polynomial in the size of $\mathcal{R}$ and $\mathcal{A}$ recognizing $\mathcal{R}^*\big(L(\mathcal{A})\big)$.*

Example 3. The TA $\mathcal{A}'_0 = \big(\{q_0, q_{c_1}\}, \{q_0\}, \Delta\big)$ recognizes $\big(\mathcal{R}_1^{-1}\big)^*(0)$, for the TRS $\mathcal{R}_1$ of Example 1, where Δ contains $0 \to q_0$, $c_1 \to q_{c_1}$, $s(q_0) \to q_{c_1}$, $s(q_{c_1}) \to q_0$, $q_0 + q_0 \to q_0$, $q_{c_1} + q_{c_1} \to q_0$, and $q + q' \to q_{c_1}$ with $q, q' \in \{q_0, q_{c_1}\}$, $q \neq q'$. $\diamond$

Now we shall use the above results for the decision of the sufficient condition for UN given in Proposition 4. Assume given a signature Σ and let us extend it with new m constant symbols $x_1, \ldots, x_m$ representing the variables in the Definition 3 of fork of languages, where m is the maximum arity of symbols in Σ. The extended signature is denoted Σ_{ext}. We consider from now on the fork of languages of Definition 3 as pairs of subsets of $\mathcal{T}(\Sigma_{\mathsf{ext}})$.

Lemma 6. *For each fork of languages $\langle L, L'\rangle$ wrt $\mathcal{R}$, L and L' are recognized by two TA over Σ_{ext} whose respective sizes are polynomial in the size of $\mathcal{R}$.*

Proof. For each $x_i \in \Sigma_{\mathsf{ext}}$ $(i \leq a)$, we construct one TA $\mathcal{A}_{x_i} = (Q_{x_i}, \{q^{\mathsf{f}}_{x_i}\}, \Delta_{x_i})$ over Σ_{ext} (with one state and one transition rule) such that $L(\mathcal{A}_{x_i}) = \{x_i\}$, and for each constant $c \in \Sigma_0$, let us construct one TA $\mathcal{A}_c = (Q_c, \{q^{\mathsf{f}}_c\}, \Delta_c)$ over Σ_{ext} such that $L(\mathcal{A}_c) = \{c\}$, and one other TA $\mathcal{A}'_c = (Q'_c, \{p^{\mathsf{f}}_c\}, \Delta'_c)$ over Σ_{ext} such that $L(\mathcal{A}'_c) = (\mathcal{R}^{-1})^*\big(\{c\}\big)$. According to Lemma 5, all the above TA are polynomial in the size of $\mathcal{R}$. Note that we assume *wlog* (this can be obtained by adding a polynomial number of ε-transitions) that all their final state sets are singleton sets. We also assume *wlog* that their state sets are pairwise disjoint.

Let $\langle f(L_1, \ldots, L_n), f(L'_1, \ldots, L'_n)\rangle$ be a fork of languages as in definition 3. Let $\mathcal{A}$ be a TA with state set $\bigcup_{i\leq m} Q_{x_i} \cup \bigcup_{c\in\Sigma_0}(Q_c \cup Q'_c) \cup \{q^{\mathsf{f}}\}$, unique final state q^{f} and whose transition set contains the rules of $\bigcup_{i\leq m} \Delta_{x_i}$ and $\bigcup_{c\in\Sigma_0}(\Delta_c \cup \Delta'_c)$ plus one single rule $f(q_1, \ldots, q_n) \to q^{\mathsf{f}}$ such that for all $i \leq n$, $q_i := q^{\mathsf{f}}_{x_j}$ if $L_i = \{x_j\}$, $q_i := q^{\mathsf{f}}_c$ if $L_i = \{c\}$ with $c \in \Sigma_0$, and $q_i := p^{\mathsf{f}}_c$ if $L_i = (\mathcal{R}^{-1})^*\big(\{c\}\big)$, with $c \in \Sigma_0$.

We associate similarly a TA $\mathcal{A}'$ to $f(L'_1, \ldots, L'_n)\rangle$. By construction, we have immediately that $\mathcal{A}$ and $\mathcal{A}'$ recognize respectively the left and right component of the given fork of language. $\square$

We have now all the ingredients to prove the main result of the paper.

Theorem 1. *UN is decidable in polynomial time for flat and linear TRS.*

Proof. Let $\mathcal{R}$ be a flat and linear TRS over Σ (hence it can be seen as a TRS over Σ_{ext}). We construct first a TA $\mathcal{A}_{\mathcal{R}}$ on Σ_{ext} recognizing $\mathsf{NF}_{\mathcal{R}} \cap \mathcal{T}(\Sigma_{\mathsf{ext}})$. According to Lemma 4, $\mathcal{A}_{\mathcal{R}}$ is polynomial in the size of $\mathcal{R}$.

Now, for each fork of languages $\langle L, L'\rangle$ wrt $\mathcal{R}$, we perform the following test. Let $\mathcal{A}$ and $\mathcal{A}'$ be two TA of size polynomial in the size of $\mathcal{R}$, recognizing respectively L and L' (constructed according to Lemma 6).

1. Construct two TA recognizing respectively $\mathcal{R}^*(L)$ and $\mathcal{R}^*(L')$. According to Lemma 5, their sizes are polynomial in the size of $\mathcal{R}$.
2. Construct two TA recognizing respectively $\mathcal{R}^*(L) \cap \mathsf{NF}_{\mathcal{R}}$ and $\mathcal{R}^*(L') \cap \mathsf{NF}_{\mathcal{R}}$, using the TA constructed at the above step and the above $\mathcal{A}_{\mathcal{R}}$. According to Proposition 1, the sizes of these two TA are still polynomial in the size of $\mathcal{R}$.

3. If one of the languages $\mathcal{R}^*(L) \cap \mathsf{NF}_\mathcal{R}$ or $\mathcal{R}^*(L') \cap \mathsf{NF}_\mathcal{R}$ is empty (this emptiness test is performed in polynomial time, according to Proposition 2) then the test passes successfully.
4. Otherwise, check whether both languages $\mathcal{R}^*(L) \cap \mathsf{NF}_\mathcal{R}$ and $\mathcal{R}^*(L') \cap \mathsf{NF}_\mathcal{R}$ are singleton sets (this test is performed in polynomial time according to Proposition 2). If it not the case, the test fails, otherwise, we construct a TA recognizing their intersection and the test succeed iff its language is not empty (this can still be checked in polynomial time).

According to Proposition 4, $\mathcal{R}$ is UN iff all fork languages of $\mathcal{R}$ pass the test. According to Lemma 3, there will be a polynomial number of such tests, and following the above evaluation, each test is performed in polynomial time. Altogether, this gives the polynomial complexity upper bound for decision of UN for flat and linear TRS. □

3 Decidability of UN for Shallow and Linear TRS

We show in this section that the result of Theorem 1 can be generalized to shallow and linear TRS, with the following reduction of shallowness into flatness.

Given $u \in \mathcal{T}(\Sigma, \mathcal{V})$, $t \in \mathcal{T}(\Sigma)$ and a constant symbol $c \notin \Sigma$, we denote $u[t \to c]$ (resp. $u[c \to t]$) the term obtained from u by the replacement of all the occurrences of t by c (resp. c by t). We extend these notations to substitutions in the natural way, i.e. for any substitution σ, $x(\sigma[t \to c]) = (x\sigma)[t \to c]$.

The iteration of the following transformations convert a shallow TRS into a flat TRS on an extended signature, replacing deep subterms by new constants.

1. if there exists a non-constant ground term t occurring as a strict subterm of a rhs of a rule, then create a new constant c, replace every rule $\ell \to r$ by $\ell \to (r[t \to c])$ and add the rule $c \to t$.
2. if there exists a non-constant ground term t occurring as a strict subterm of a lhs of a rule, then create a new constant c, replace every rule $\ell \to r$ by $\ell[t \to c] \to r$ and add the rule $t \to c$. Moreover, if t was not a normal form before this change, then add also $c \to c$.

Example 4. $\mathcal{R} = \{x + 0 \to x, x + s(0) \to s(x), x + y \to y + x, s(s(0)) \to 0\}$ is a shallow and linear TRS for the commutative addition of positive integers modulo 2. The above flattening transformations terminates on $\mathcal{R}$ after one step of application of the case 2, returning the flat and linear TRS of Example 1. ◇

Proposition 5. *Given a shallow and linear TRS $\mathcal{R}$ on Σ, there exists a flat and linear TRS $\mathcal{R}'$ on an extended signature $\Sigma' \supseteq \Sigma$ such that $\mathcal{R}'$ is UN iff $\mathcal{R}$ is UN. Moreover, the size of $\mathcal{R}'$ is polynomial in the size of $\mathcal{R}$.*

Proof. Let $\mathcal{R}_0 = \mathcal{R}, \mathcal{R}_1, \ldots$ be the TRS obtained by the repeated application of the above transformations and $\Sigma_0 = \Sigma \subseteq \Sigma_1 \subseteq \ldots$ be the signatures associated, *i.e.* $\Sigma_{i+1} = \Sigma_i \uplus \{c\}$. Note that the iteration of the transformations terminates after a number of steps n linear in the size of $\mathcal{R}$, with a TRS $\mathcal{R}' = \mathcal{R}_n$ over

$\Sigma' = \Sigma_n$ (Σ' contains Σ and the new constants c) whose size is polynomial in the size of $\mathcal{R}$. Note also that every $\mathcal{R}_i$ is shallow and linear and that $\mathcal{R}'$ is flat and linear. We shall use the following fact below.

Fact 1 *For all $i < n$ and $u, v \in \mathcal{T}(\Sigma_{i+1}, \mathcal{V})$, if $u \xrightarrow[\mathcal{R}_{i+1}]{*} v$ then $u[c \to t] \xrightarrow[\mathcal{R}_i]{*} v[c \to t]$.*

Proof. By induction on the length of $u \xrightarrow[\mathcal{R}_{i+1}]{*} v$. The base case is immediate. For the induction step, assume that $u \xrightarrow[\mathcal{R}_{i+1}]{*} v' \xrightarrow[\mathcal{R}_{i+1}]{} v$ and that $u[c \to t] \xrightarrow[\mathcal{R}_i]{*} v'[c \to t]$. If the last rewrite step $v' \xrightarrow[\mathcal{R}_{i+1}]{} v$ involves a rule of $\mathcal{R}_i$ then we are done, because by construction the rules of $\mathcal{R}_i$ do not contain c. Otherwise, we need to consider the two above cases of construction of the rule used.

Case 1. If the last rewrite step $v' \xrightarrow[\mathcal{R}_{i+1}]{} v$ involves the rule $c \to t$, then $v[c \to t] = v'[c \to t]$ (by construction, t does not contain the symbol c) and we are done. If it involves a rule $\ell \to r[t \to c]$, with $\ell \to r \in \mathcal{R}_i$, at a position p of v' and with a matcher σ, then $v' = v'[\ell\sigma]_p$ and $v = v'[r[t \to c]\sigma]_p$. By construction, the symbol c does not occur in ℓ and r, hence $v'[c \to t] = v'[c \to t]\big[\ell(\sigma[c \to t])\big]_p$ and $v[c \to t] = v'[c \to t]\big[r(\sigma[c \to t])\big]_p$. It means that $u[c \to t] \xrightarrow[\mathcal{R}_i]{} v[c \to t]$.

Case 2. If the last rewrite step $v' \xrightarrow[\mathcal{R}_{i+1}]{} v$ involves the rule $t \to c$, then again $v[c \to t] = v'[c \to t]$. If it involves a rule $\ell[t \to c] \to r$, with $\ell \to r \in \mathcal{R}_i$, at a position p of v' and with a matcher σ, then $v' = v'\big[\ell[t \to c]\sigma\big]_p$ and $v = v'[r\sigma]_p$. Since c does not occur in ℓ and r, we have $v'[c \to t] = v'[c \to t][\ell\sigma]_p$ and it follows that $u[c \to t] \xrightarrow[\mathcal{R}_i]{} v[c \to t]$. □

We show now that for all $i < n$, $\mathcal{R}_i$ is UN iff $\mathcal{R}_{i+1}$ is UN. It will permit to conclude. Let c be the new constant added at step $i+1$, and t be the corresponding term. We consider the two possible transformation steps.

Case 1. Assume that $\mathcal{R}_{i+1}$ is not UN, *i.e.* that there exists $u, v_1, v_2 \in \mathcal{T}(\Sigma_{i+1}, \mathcal{V})$, such that $v_1 \xleftarrow[\mathcal{R}_{i+1}]{*} u \xrightarrow[\mathcal{R}_{i+1}]{*} v_2$ and v_1, v_2 are $\mathcal{R}_{i+1}$-normal-forms. From Fact 1, it follows that $u[c \to t] \xrightarrow[\mathcal{R}_i]{*} v_j[c \to t]$ for $j = 1, 2$, and hence, that $\mathcal{R}_i$ is not UN, because $v_j[c \to t] = v_j$ for $j = 1, 2$ ($\mathcal{R}_{i+1}$-normal-forms do not contain c).

For the other direction, it is sufficient to observe that all $\mathcal{R}_i$-normal-forms are $\mathcal{R}_{i+1}$-normal-forms and that $\xrightarrow[\mathcal{R}_i]{} \subseteq \xrightarrow[\mathcal{R}_{i+1}]{*}$. Indeed, if $v_1 \xleftarrow[\mathcal{R}_i]{*} u \xrightarrow[\mathcal{R}_i]{*} v_2$ for $u, v_1, v_2 \in \mathcal{T}(\Sigma_i, \mathcal{V})$ and v_1, v_2 are $\mathcal{R}_i$-normal-forms, then if $v_1 \xleftarrow[\mathcal{R}_{i+1}]{*} u \xrightarrow[\mathcal{R}_{i+1}]{*} v_2$ and v_1, v_2 are also $\mathcal{R}_{i+1}$-normal-forms, *i.e.* $\mathcal{R}_{i+1}$ is not UN.

Case 2. If t is not a $\mathcal{R}_i$-normal form, then $\mathcal{R}_{i+1}$ contains $c \to c$. In this case the $\mathcal{R}_{i+1}$-normal-forms do not contain c, and every $\mathcal{R}_i$-normal-forms is a $\mathcal{R}_{i+1}$-normal-forms (because it does neither contain c nor t). Moreover, $\xrightarrow[\mathcal{R}_i]{} \subseteq \xrightarrow[\mathcal{R}_{i+1}]{*}$. Therefore, we can do the proof in the same lines as above for the case 1.

If t is a $\mathcal{R}_i$-normal form, we do the following observations.

Fact 2 *i. If $u \in \mathcal{T}(\Sigma_{i+1}, \mathcal{V})$ is a $\mathcal{R}_{i+1}$-NF then $u[c \to t]$ is a $\mathcal{R}_i$-NF.*
ii. If $u \in \mathcal{T}(\Sigma_i, \mathcal{V})$ is a $\mathcal{R}_i$-NF then $u[t \to c]$ is a $\mathcal{R}_{i+1}$-NF.

Proof. (*i.*) Let $u \in \mathcal{T}(\Sigma_{i+1}, \mathcal{V})$ be such that $u[c \to t]$ is $\mathcal{R}_i$-reducible, *i.e.* such that there exists a rule $\ell \to r \in \mathcal{R}_i$, a position $p \in \mathcal{P}os(u)$ and a substitution σ such that $u[c \to t]|_p = \ell\sigma$. Note that $p \in \mathcal{P}os(u)$ and $u|_p \neq c$ because t is supposed to be a $\mathcal{R}_i$-normal form. If t is a subterm of $u|_p$, then $u|_p$ is $\mathcal{R}_{i+1}$-reducible with the rule $t \to c$. Otherwise, $u|_p = \big(u[c \to t]|_p\big)[t \to c] = (\ell\sigma)[t \to c] = \big(\ell[t \to c]\big)(\sigma[t \to c])$ because $\mathcal{R}_i$ is shallow and linear and ℓ does not match t (which is a $\mathcal{R}_i$-NF by hypothesis). Hence $u|_p$ (and also u) is $\mathcal{R}_{i+1}$-reducible with the rule $\ell[t \to c] \to r$.

(*ii.*) Let $u \in \mathcal{T}(\Sigma_i, \mathcal{V})$ be such that $u[t \to c]$ is $\mathcal{R}_{i+1}$-reducible. Note that $u[t \to c]$ cannot be reducible by $t \to c$. Thus, $u[t \to c]$ is reducible by a rule $\ell[t \to c] \to r \in \mathcal{R}_{i+1}$. Let $p \in \mathcal{P}os(u)$ and σ be such that $u[t \to c]|_p = \ell[t \to c]\sigma$. We have $u|_p = \ell(\sigma[c \to t])$, and hence u is $\mathcal{R}_i$-reducible with the rule $\ell \to r$. □

Assume that $\mathcal{R}_{i+1}$ is not UN. Then there exist $u, v_1, v_2 \in \mathcal{T}(\Sigma_{i+1}, \mathcal{V})$, such that v_1, v_2 are $\mathcal{R}_{i+1}$-NF and $v_1 \xleftarrow[\mathcal{R}_{i+1}]{*} u \xrightarrow[\mathcal{R}_{i+1}]{*} v_2$. It implies that $u[c \to t] \xrightarrow[\mathcal{R}_i]{*} v_j[c \to t]$ for $j = 1, 2$ by Fact 1, and hence, $\mathcal{R}_i$ is not UN by Fact 2.*i*. Note that $v_1[c \to t] \neq v_2[c \to t]$, because v_1 and v_2 do not contain t, and hence $v_1 = v_1[c \to t][t \to c]$ and $v_2 = v_2[c \to t][t \to c]$, and $v_1 \neq v_2$t.

For the other direction, assume that $\mathcal{R}_i$ is not UN. Then, there exist $u, v_1, v_2 \in \mathcal{T}(\Sigma_i, \mathcal{V})$, such that v_1, v_2 are $\mathcal{R}_i$-normal forms and $u \xrightarrow[\mathcal{R}_i]{*} v_j$, $j = 1, 2$. It follows that $u \xrightarrow[\mathcal{R}_{i+1}]{*} v_j \xrightarrow[t \to c]{*} v_j[t \to c]$ for $j = 1, 2$ ($\xrightarrow[\mathcal{R}_i]{*} \subseteq \xrightarrow[\mathcal{R}_{i+1}]{*}$), hence $\mathcal{R}_{i+1}$ is not UN by Fact 2.*ii*. Again, note that $v_1[t \to c] \neq v_2[t \to c]$ are different, because v_1 and v_2 do not contain c, and hence, $v_1 = v_1[t \to c][c \to t]$ and $v_2 = v_2[t \to c][c \to t]$, and v_1 and v_2 are different. □

From Theorem 1 and Proposition 5 it follows that

Corollary 1. *UN is decidable in polynomial time for shallow and linear TRS.*

4 Undecidability of UN for Flat and Right-Linear TRS

The decision result of Theorem 1 is no longer valid if we relax the assumptions on flatness and linearity of the TRS. Indeed, if we keep the assumption on flatness but relax linearity so that only the right-hand side must be linear, then UN becomes undecidable.

Theorem 2. *UN is undecidable for flat and right-linear TRS.*

The proof involves a reduction from the Post correspondence problem (PCP) restricted to nonempty strings over a fixed finite alphabet Γ, i.e.:

Problem: Restricted-PCP.
Instance: A sequence of pairs of words $\langle u_1, v_1\rangle \dots \langle u_n, v_n\rangle$, with $\forall i \leq n$, $u_i, v_i \in \Gamma^* \setminus \Lambda$.
Question: Is there a non-empty sequence of indexes $1 \leq i_1, \dots, i_k \leq n$ such that $u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$?

A *solution* for a PCP instance is a list of indexes that gives a positive answer to the previous question.

Example 5. The instance of PCP $\langle ab, a\rangle, \langle c, bc\rangle$ has a solution obtained by choosing the pairs 1 and 2 consecutively, obtaining abc at both sides. $\diamond$

Proposition 6 ([14]). *The restricted-PCP is undecidable.*

Along the next pages we assume given an instance $\langle u_1, v_1\rangle \dots \langle u_n, v_n\rangle$ of the restricted-PCP, that is, u_i, v_i are nonempty strings over alphabet Γ.

We define a TRS $\mathcal{R}$ such that the given PCP instance has a solution iff $\mathcal{R}$ is not uniquely normalizing. Let $L = \max(|u_1|, \dots, |u_n|, |v_1|, \dots, |v_n|)$. Let us first define the signature $\Sigma = \Sigma_0 \uplus \Sigma_1 \uplus \Sigma_8$ ($[m]$ denotes the segment $[1..m]$):

$$\begin{aligned}
\Sigma_0 &:= \{U, U', U'', V, V', V'', P, P', A, A', 0, 1\} \cup \\
&\qquad \big\{U'_{ij}, V'_{ij} \;\big|\; i \in [n], j \in [L+1]\big\} \cup \big\{P'_j \;\big|\; j \in [0..L+1]\big\} \\
\Sigma_1 &:= \Gamma \cup \big\{U_{ij}, U''_{ij}, V_{ij}, V''_{ij}, P_{ij} \;\big|\; i \in [n], j \in [L]\big\} \\
\Sigma_8 &:= \{f\}
\end{aligned}$$

The j'th symbol of u_i and v_i, whenever it exists, is denoted by u_{ij} and v_{ij} respectively. For sake of readability, we make no distinction below between terms of $\mathcal{T}(\Sigma_1 \cup \Sigma_0, \mathcal{V})$ and words of $\Sigma_1^*(\Sigma_0 \cup \mathcal{V})$. The TRS $\mathcal{R}$ is the union of the following sets of flat and right-linear rules.

$$\begin{aligned}
\mathcal{R}_{U'} &:= \big\{U' \to U'_{i1} \mid i \in [n]\big\} \cup \big\{U'_{ij} \to U''_{ij}U'_{i(j+1)} \;\big|\; i \in [n], j \in [L]\big\} \cup \\
&\qquad \big\{U'_{i(L+1)} \to U',\ U'_{i(L+1)} \to U'' \;\big|\; i \in [1..n]\big\} \\
\mathcal{R}_{V'} &:= \big\{V' \to V'_{i1} \;\big|\; i \in [n]\big\} \cup \big\{V'_{ij} \to V''_{ij}V'_{i(j+1)} \;\big|\; i \in [n], j \in [L]\big\} \cup \\
&\qquad \big\{V'_{i(L+1)} \to V'',\ V'_{i(L+1)} \to V' \;\big|\; i \in [n]\big\} \\
\mathcal{R}_{A'} &:= \{A' \to \gamma(A') \mid \gamma \in \Gamma\} \\
\mathcal{R}_{P'} &:= \big\{P' \to P'_{i1},\ P'_{ij} \to P_{ij}P'_{i(j+1)},\ P'_{i(L+1)} \to P' \;\big|\; i \in [n], j \in [L]\big\} \\
\hline
\mathcal{R}_{UV} &:= \big\{U_{ij}x \to U''_{ij}x,\ V_{ij}x \to V''_{ij}x \;\big|\; i \in [n], j \in [L]\big\} \\
\mathcal{R}_{A} &:= \big\{U_{ij}x \to u_{ij}x \;\big|\; j \in \big[1..|u_i|\big]\big\} \cup \big\{U_{ij}x \to x \;\big|\; j \in \big[|u_i|+1..L\big]\big\} \cup \\
&\qquad \big\{V_{ij}x \to v_{ij}x \;\big|\; j \in \big[1..|v_i|\big]\big\} \cup \big\{V_{ij}x \to x \;\big|\; j \in \big[|v_i|+1..L\big]\big\} \\
\mathcal{R}_{P} &:= \big\{U_{ij}x \to P_{ij}x, V_{ij}x \to P_{ij}x \;\big|\; i \in [n], j \in [L]\big\} \\
\hline
\mathcal{R}_{Q} &:= \{U \to P, U \to A, U \to U'', V \to P, V \to A, V \to V'', A' \to A, P' \to P\} \\
\mathcal{R}_{f} &:= \{f(U', V', x, y, x, y, x, y) \to 0,\ f(x, y, A', P', x, x, y, y) \to 1\} \\
\hline
\mathcal{R}_{0} &:= \big\{c \to c \mid c \in \Sigma_0 \setminus \{0, 1\}\big\} \\
\mathcal{R}_{1} &:= \{Xy \to Xy \mid X \in \Sigma_1\} \\
\mathcal{R}_{8} &:= \{f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \to f(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)\}
\end{aligned}$$

The subsystems $\mathcal{R}_{U'}$, $\mathcal{R}_{V'}$, $\mathcal{R}_{P'}$ and $\mathcal{R}_{A'}$ are *generators*. The three first generate sequences of blocks of the form $U''_{i1} \dots U''_{iL}$ for $\mathcal{R}_{U'}$, with $i \in [n]$ (resp. $V''_{i1} \dots V''_{iL}$ and $P_{i1} \dots P_{iL}$ for $\mathcal{R}_{V'}$, and $\mathcal{R}_{P'}$) starting from U' (resp. V', P'). The TRS $\mathcal{R}_{A'}$ generate words of $\Gamma^* A'$ starting from A'.

The subsystems $\mathcal{R}_{UV}$, $\mathcal{R}_A$ and $\mathcal{R}_P$ are *converters*: they cast a block of the form $U_{i1} \dots U_{iL}$ into respectively $U''_{i1} \dots U''_{iL}$, the word u_i and $P_{i1} \dots P_{iL}$ (and

analogously, a block of the form $V_{i1} \dots V_{iL}$ is casted into $V''_{i1} \dots V''_{iL}$, v_i and $P_{i1} \dots P_{iL}$). The rules of $\mathcal{R}_Q$ and $\mathcal{R}_f$ will act as *checkers* for the correctness of the solution. The rules of $\mathcal{R}_0$, $\mathcal{R}_1$ and $\mathcal{R}_8$ are just to ensure that no term is a normal form except for 0, 1 and variables. The following lemma shows how one can construct a term with two reachable normal forms from a solution for the given PCP instance.

Lemma 7. *If the given restricted-PCP instance $\langle u_1, v_1\rangle \dots \langle u_n, v_n\rangle$ has a solution, then there exists a term $s \in \mathcal{T}(\Sigma)$ such that $0 \xleftarrow{*}_{\mathcal{R}} s \xrightarrow{*}_{\mathcal{R}} 1$.*

Proof. Let $i_1, \dots, i_k$ be a solution of the PCP instance, let $w = u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$, and let s_U, s''_U, s_V, s''_V, s_P and s be defined as follows.

$$\begin{array}{ll} s_U := U_{i_1 1} \dots U_{i_1 L} \ \dots \ U_{i_k 1} \dots U_{i_k L} U & s_V := V_{i_1 1} \dots V_{i_1 L} \ \dots \ V_{i_k 1} \dots V_{i_k L} V \\ s''_U := U''_{i_1 1} \dots U''_{i_1 L} \ \dots \ U''_{i_k 1} \dots U''_{i_k L} U'' & s''_V := V''_{i_1 1} \dots V''_{i_1 L} \ \dots \ V''_{i_k 1} \dots V''_{i_k L} V'' \\ s_P := P_{i_1 1} \dots P_{i_1 L} \ \dots \ P_{i_k 1} \dots P_{i_k L} P & s := f(U', V', A', P', s_U, s_U, s_V, s_V) \end{array}$$

It is easy to verify that $U' \xrightarrow[\mathcal{R}_{U'}]{*} s''_U \xleftarrow[\mathcal{R}_{UV}]{*} s_U$ and $V' \xrightarrow[\mathcal{R}_{V'}]{*} s''_V \xleftarrow[\mathcal{R}_{UV}]{*} s_V$. Similarly, A', S_U and S_V can be rewritten to wA using $\mathcal{R}_{A'}$, $\mathcal{R}_A$ and $\mathcal{R}_Q$. Similarly also, P', S_U and S_V can be rewritten to s_P using $\mathcal{R}_{P'}$, $\mathcal{R}_P$ and $\mathcal{R}_Q$. Therefore, there exist derivations $s \xrightarrow[\mathcal{R}]{*} f(U', V', w, s_P, w, s_P, w, s_P) \xrightarrow[\mathcal{R}_f]{} 0$ and $s \xrightarrow[\mathcal{R}]{*} f(s''_U, s''_V, A', P', s''_U, s''_U, s''_V, s''_V) \xrightarrow[\mathcal{R}_f]{} 1$ and this concludes the proof. □

It remains to see that non-unique normalization implies the existence of a solution for the given PCP instance. This requires more detailed arguments. Given a word $w \in \Sigma_1^*$, we define $idx(w)$ to be the integer sequence obtained by application to w of the morphism ϕ defined by $\phi(U_{i1}) = \phi(U''_{i1}) = \phi(V_{i1}) = \phi(V''_{i1}) = \phi(P_{i1}) = i$ and $\phi(X) = \Lambda$ for any other symbol X. We define Σ_c as the subset of Σ_1 of the symbols h for which there exists a collapsing rule $hx \to x$ in $\mathcal{R}$, i.e. Σ_c contains all U_{ij} such that $j > |u_i|$, and all V_{ij} such that $j > |v_i|$.

The following lemma follows directly from the form of the rules in $\mathcal{R}$ and the fact that $|u_i|, |v_i| \neq 0$ for all i.

Lemma 8. *For all $X \in \{U', V', A', P'\}$, and word s, if $s \xrightarrow{*}_{\mathcal{R}} X$ then $s \in \Sigma_c^* X$.*

A word wX is called a *generator* if $X \in \{U', V', A', P'\}$.

Lemma 9. *The word wX is not a generator if it is joinable by $\mathcal{R}$ either with a word in $\Sigma_c^* U'$ and (separately) with a word in $\Sigma_c^* A'$, or with a word in $\Sigma_c^* V'$ and a word in $\Sigma_c^* A'$, or with a word in $\Sigma_c^* U'$ and a word in $\Sigma_c^* P'$, or with a word in $\Sigma_c^* V'$ and a word in $\Sigma_c^* P'$.*

Lemma 10. *If $w_1 X_1$ and $w_2 X_2$ are not generators, $w_3 X_3 \in \Sigma_c^* U' \cup \Sigma_c^* V' \cup \Sigma_c^* P'$ and $\{w_1 X_1, w_2 X_2, w_3 X_3\}$ are joinable by $\mathcal{R}$, then $idx(w_1) = idx(w_2)$.*

Proof. Note that no word in $\Sigma_c^* U'$ nor in $\Sigma_c^* V'$ nor in $\Sigma_c^* P'$ can be rewritten by $\mathcal{R}$ to a word containing some symbol in Γ. Hence, when joining $\{w_1 X_1, w_2 X_2, w_3 X_3\}$ with certain derivations from them, no rule of the form $U_{i1} x \to u_{i1} x$ nor $V_{i1} x \to v_{i1} x$ is used, since any of such u_{i1} or v_{i1} is not Λ. Therefore, idx is preserved by rewriting for all the words occurring in the considered derivations, and all of them reach the same term. □

Lemma 11. *If w_1X_1 is not a generator, is joinable with a word in Σ_c^*U' (resp. with a word in Σ_c^*V'), $w_1X_1 \xrightarrow[\mathcal{R}]{*} w_2w_3A$ where $w_3 \in \Gamma^*$ and $w_2 \in \left(\Sigma_c \cup \{P_{ij} \mid j > |u_i|\}\right)^*$ (resp. $w_2 \in \left(\Sigma_c \cup \{P_{ij} \mid j > |v_i|\}\right)^*$), and $i_1, \ldots, i_k = idx(w_1)$, then $w_3 = u_{i_1} \ldots u_{i_k}$ (resp. $w_3 = v_{i_1} \ldots v_{i_k}$).*

Proof. Note that a word w_4X_4 in Σ_c^*U' cannot reach with $\mathcal{R}$ a word containing any symbol in Γ. Hence, if w_1X_1 and w_4X_4 are joinable by $\mathcal{R}$, then no non-collapsing rule of the form $U_{ij}x \to u_{ij}x$ is used in the reductions. For joining with w_4X_4, but also for reaching w_2w_3A, the word w_1X_1 has to be of the form $w_1'w_1''U$, such that w_1' contains symbols of Σ_c or symbols P_{ij} such that $j > |u_i|$ and moreover, if we remove any symbol of Σ_c from w_1'', the result is the word $U_{i_11} \ldots U_{i_1|u_{i_1}|} \ldots U_{i_k1} \ldots U_{i_k|u_{i_k}|}$. It follows that $w_3 = u_{i_1} \ldots u_{i_k}$. The proof of the case with V' and v_i is similar. □

Lemma 12. *If $\mathcal{R}$ is not UN then the given restricted-PCP instance $\langle u_1, v_1\rangle \ldots \langle u_n, v_n\rangle$ has a solution.*

Proof. Let s be a term in $\mathcal{T}(\Sigma, \mathcal{V})$, minimal in size, such that $s_0 \xleftarrow[\mathcal{R}]{*} s \xrightarrow[\mathcal{R}]{*} s_1$ where s_0 and s_1 are distinct $\mathcal{R}$-normal forms. Note that only 0, 1 and variables are $\mathcal{R}$-normal forms.

The term s is not headed by a symbol in $\Sigma \setminus (\Sigma_c \cup \{f, 0, 1\})$, otherwise it could not reach a $\mathcal{R}$-normal form, because of the rules of $\mathcal{R}_0 \cup \mathcal{R}_1$ and because such symbols cannot be removed.

Assume that s is headed by a symbol h in Σ_c, i.e. s is of the form hs'. Then in both derivations $s \xrightarrow[\mathcal{R}]{*} s_0$ and $s \xrightarrow[\mathcal{R}]{*} s_1$, the rule $hx \to x$ is applied at the root position. Hence, the term s' also reaches s_0 and s_1 with $\mathcal{R}$, contradicting the minimality of s.

It follows that s is headed by f. Let s be of the form $f(s_U', s_V', s_A', s_P', s_U, t_U, s_V, t_V)$ for some terms $s_U', s_V', s_A', s_P', s_U, t_U, s_V, t_V$. Since no variable is reachable from a term of this form (there are no collapsing rules with a f at the top of lhs), we conclude that $s_0 = 0$ and $s_1 = 1$. Thus, the rules $f(U', V', x, y, x, y, x, y) \to 0$, $f(x, y, A', P', x, x, y, y) \to 1$ are applied at the root position in the derivations from s. This implies the following facts.

- $s_U' \xrightarrow[\mathcal{R}]{*} U'$, and hence, by Lemma 8, $s_U' \in \Sigma_c^*U'$. Similarly, $s_V' \in \Sigma_c^*V'$, $s_A' \in \Sigma_c^*A'$, and $s_P' \in \Sigma_c^*P'$.
- s_U', s_U are joinable with $\mathcal{R}$, and s_U, s_A' are joinable too. Hence, by Lemma 9, s_U is not a generator. Similarly, t_U, s_V and t_V are not generators.
- $\{s_U', s_U, t_U\}$ are joinable by $\mathcal{R}$, and hence, by Lemma 10, $idx(s_U) = idx(t_U)$. Similarly, $\{s_V', s_V, t_V\}$ are joinable and $idx(s_V) = idx(t_V)$; and $\{s_P', t_U, t_V\}$ are joinable and $idx(t_U) = idx(t_V)$. Let $i_1, \ldots, i_k$ be idx of any of them.
- $\{s_U', s_U\}$ are joinable by $\mathcal{R}$, and $\{s_A', s_U, s_V\}$ are joinable to a word of the form w_2w_3A such that $w_3 \in \Sigma^*$ and $w_2 \in (\Sigma_c \cup \{P_{ij} | j > |u_i|\})^*$. Hence, by Lemma 11, $w_3 = u_{i_1} \ldots u_{i_k}$. Similarly, $\{s_V', s_V\}$ are joinable, and $s_V \xrightarrow[\mathcal{R}]{*} w_2w_3A$, for the same w_2 and w_3 as before, and hence, $w_3 = v_{i_1} \ldots v_{i_k}$.

From the previous facts, it follows that $i_1, \ldots, i_k$ is a solution of the given restricted-PCP instance. □

References

1. Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.
2. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on http://tata.gforge.inria.fr, 2007.
3. H. Comon and F. Jacquemard. Ground reducibility is EXPTIME-complete. *Information and Computation*, 187(1):123–153, 2003.
4. G. Godoy and H. Hernández. Undecidable properties for flat term rewrite systems. *Submitted*, 2007.
5. G. Godoy, E. Huntingford, and A.Tiwari. Termination of rewriting with right-flat rules. In F. Baader, editor, *18th Int. Conf. on Term Rewriting and Applications, (RTA)*, vol. 4533 of *LNCS*, pages 200–213. Springer, 2007.
6. G. Godoy and S. Tison. On the normalization and unique normalization properties of term rewrite systems. In *Proc. 21st International Conference on Automated Deduction (CADE)*, vol. 4603 of *LNCS*, pages 247–262, 2007.
7. G. Godoy and A. Tiwari. Deciding fundamental properties of right-(ground or variable) rewrite systems by rewrite closure. In *Intl. Joint Conf. on Automated Deduction (IJCAR)*, vol. 3097 of *LNAI*, pages 91–106. Springer, 2004.
8. G. Godoy and A. Tiwari. Confluence of shallow right-linear rewrite systems. In *19th International Workshop of Computer Science Logic (CSL)*, vol. 3634 of *LNCS*, pages 541–556. Springer, 2005.
9. F. Jacquemard. Decidable approximations of term rewriting systems. In *7th Int. Conf. Rewriting Techniques and Applications (RTA)*, *LNCS*, pages 362–376. Springer, 1996.
10. F. Jacquemard. Reachability and confluence are undecidable for flat term rewriting systems. *Inf. Process. Lett.*, 87(5):265–270, 2003.
11. I. Mitsuhashi, M. Oyamaguchi, and F. Jacquemard. The confluence problem for flat TRSs. In *Proc. 8th Intl. Conf. on Artificial Intelligence and Symbolic Computation (AISC)*, vol. 4120 of *LNAI*, pages 68–81. Springer, 2006.
12. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *Inf. Comput.*, 178(2):499–514, 2002.
13. K. Salomaa. Deterministic tree pushdown automata and monadic tree rewriting systems. *J. Comput. Syst. Sci.*, 37:367–394, 1998.
14. M. Sipser. *Introduction to the Theory of Computation*. Course Technology, 2006.
15. T. Takai, Y. Kaji, and H. Seki. Right-linear finite path overlapping term rewriting systems effectively preserve recognizability. In *Rewriting Techniques and Applications (RTA)*, vol. 1833 of *LNCS*, pages 246–260, 2000.
16. R. Verma. Complexity of normal form properties and reductions for rewriting problems. *Fundamenta Informaticae*. Accepted for publication.
17. R. Verma and A. Hayrapetyan. A new decidability technique for ground term rewriting systems. *ACM Trans. Comput. Log.*, 6(1):102–123, Dec 2005.
18. Y. Wang and M. Sakai. Decidability of termination for semi-constructor trss, left-linear shallow trss and related systems. In *Proc. 17th Intl. Conf. RTA*, vol. 4098 of *LNCS*, pages 343–356. Springer, 2006.

A Remaining Proofs of Lemmas 1 and 2

We give the proofs of the two technical lemmas of Section 2.1.

Lemma 1. *Given a derivation $s \xrightarrow[\mathcal{R}]{*} t$, a position $p \in \mathcal{P}os(s)$ and a constant c, if $use(p, s \xrightarrow[\mathcal{R}]{*} t) = c$, then $s[c]_p \xrightarrow[\mathcal{R}]{*} t$ and $s|_p \xrightarrow[\mathcal{R}]{*} c$.*

Proof. This is easily proved by induction on the length of $s \xrightarrow[\mathcal{R}]{*} t$ and the definition of *use*.

If $s|_p$ is a constant, then, by the definition of *use*, $s|_p$ is precisely c, and the result follows trivially.

If $s|_p$ is not a constant and $s \xrightarrow[\mathcal{R}]{*} t$ is the empty sequence, then $use(s \xrightarrow[\mathcal{R}]{*} t)$ is undefined, contradicting the assumptions of the lemma.

If $s|_p$ is not a constant and $s \xrightarrow[\mathcal{R}]{*} t$ is of the form $s \xrightarrow[p_1,\mathcal{R}]{} s' \xrightarrow[\mathcal{R}]{*} t$ for a position p_1 such that $p_1 \geq p$ or p_1 is disjoint with p, then $use(p, s' \xrightarrow[\mathcal{R}]{*} t) = use(p, s \xrightarrow[\mathcal{R}]{*} t) = c$. By induction hypothesis, $s'[c]_p \xrightarrow[\mathcal{R}]{*} t$ and $s'|_p \xrightarrow[\mathcal{R}]{*} c$, from which $s[c] \xrightarrow[p_1,\mathcal{R}]{} s'[c] \xrightarrow[\mathcal{R}]{*} t$ or $s[c] = s'[c] \xrightarrow[\mathcal{R}]{*} t$ follows, depending on whether p_1 is disjoint or a suffix of p. The fact that $s|_p \xrightarrow[\mathcal{R}]{*} c$ also follows, in either case.

If $s|_p$ is not a constant and $s \xrightarrow[\mathcal{R}]{*} t$ is of the form $s \xrightarrow[p_1,\ell\to r]{} s' \xrightarrow[\mathcal{R}]{*} t$, where p is of the form $p_1.i.p_2$, and $\ell|_i$ is a variable, we consider two cases. If $\ell|_i$ does not occur in r, then $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ is undefined, which contradicts the assumptions of the lemma. Otherwise, if $r|_q = \ell|_i$ for some $q \in \mathcal{P}os(r)$, then $use(p_1.q.p_2, s' \xrightarrow[\mathcal{R}]{*} t) = use(p, s \xrightarrow[\mathcal{R}]{*} t) = c$, and by induction hypothesis, $s'[c]_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} t$ and $s'|_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} c$. But then, $s[c]_p \xrightarrow[p_1,\ell\to r]{} s'[c]_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} t$ and $s|_p = s'|_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} c$ follow. □

Lemma 2. *Let $s \xrightarrow[\mathcal{R}]{} t$ be a derivation, p be a position of s such that $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ is undefined, and x be a variable. Then, either $s[x]_p \xrightarrow[\mathcal{R}]{*} t$, or there exists a position $q \in \mathcal{P}os(t)$ such that $s[x]_p \xrightarrow[\mathcal{R}]{*} t[x]_q$ and $s|_p \xrightarrow[\mathcal{R}]{*} t|_q$.*

Proof. This is easily proved by induction on the length of $s \xrightarrow[\mathcal{R}]{*} t$ and the definition of *use*.

If $s|_p$ is a constant, then $use(p, s \xrightarrow[\mathcal{R}]{*} t)$ is defined, contradicting the assumptions of the lemma.

If $s|_p$ is not a constant and $s \xrightarrow[\mathcal{R}]{*} t$ is an empty derivation, then $s = t$. By choosing $q = p$ we have $s[x]_p = t[x]_q$ and the result follows trivially.

If $s|_p$ is not a constant and $s \xrightarrow[\mathcal{R}]{*} t$ is of the form $s \xrightarrow[p_1,\mathcal{R}]{} s' \xrightarrow[\mathcal{R}]{*} t$ for a position p_1 such that $p_1 \geq p$ or p_1 is disjoint with p, then $use(p, s' \xrightarrow[\mathcal{R}]{*} t)$ is also undefined. By induction hypothesis, either $s'[x]_p \xrightarrow[\mathcal{R}]{*} t$, or there exists a position q in $\mathcal{P}os(t)$ such that $s'[x]_p \xrightarrow[\mathcal{R}]{*} t[x]_q$ and $s'|_p \xrightarrow[\mathcal{R}]{*} t|_q$. The statement of the lemma follows then for s, since either p_1 is a suffix of p, $s|_p \xrightarrow[\mathcal{R}]{*} s'|_p$ and $s[x]_p = s'[x]_p$, or p_1 is disjoint with p, $s[x]_p \xrightarrow[\mathcal{R}]{} s'[x]_p$ and $s|_p = s'|_p$.

If $s|_p$ is not a constant and $s \xrightarrow[\mathcal{R}]{*} t$ is of the form $s \xrightarrow[p_1,\ell\to r]{} s' \xrightarrow[\mathcal{R}]{*} t$, where p is of the form $p_1.i.p_2$, and $\ell|_i$ is a variable, we consider two cases. If $l|_i$ does not occur in r, then we also have $s[x]_p \xrightarrow[p_1,\ell\to r]{} s'$, and hence $s[x]_p \xrightarrow[\mathcal{R}]{*} t$. Otherwise, if $r|_q = \ell|_i$ for some $q \in \mathcal{P}os(r)$, then $use(p_1.q.p_2, s' \xrightarrow[\mathcal{R}]{*} t)$ is also undefined. By induction hypothesis, either $s'[x]_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} t$, or there exists a position q' in $\mathcal{P}os(t)$ such that $s'[x]_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} t[x]_{q'}$ and $s'|_{p_1.q.p_2} \xrightarrow[\mathcal{R}]{*} t|_{q'}$. But we also have $s[x]_p \xrightarrow[p_1,\ell\to r]{} s'[x]_{p_1.q.p_2}$. By composing this rewrite step with any of the two possible derivations, depending on the case, the result follows. □