New Undecidability Results for Properties of Term Rewrite Systems

Rakesh Verma[1,2]

Computer Science Department
University of Houston
Houston, TX
USA

Abstract

This paper is on several basic properties of term rewrite systems: reachability, joinability, uniqueness of normal forms, unique normalization, confluence, and existence of normal forms, for subclasses of rewrite systems defined by syntactic restrictions on variables. All these properties are known to be undecidable for the general class and decidable for ground (variable-free) systems. Recently, there has been impressive progress on efficient algorithms or decidability results for many of these properties. The aim of this paper is to present new results and organize existing ones to clarify further the boundary between decidability and undecidability for these properties. Another goal is to spur research towards a *complete* classification of these properties for subclasses defined by syntactic restrictions on variables. The proofs of the presented results may be intrinsically interesting as well due to their economy, which is partly based on improved reductions between some of the properties.

Keywords: Term Rewrite Systems, Decision Problems, Reachability, Confluence, Normalization Properties

1 Introduction

Programming language interpreters, proving equations, abstract data types, program transformation and optimization, and even computation itself can all be specified by a set of rules, called a rewrite system. The rules are used to replace ("reduce") subexpressions of given expressions by other expressions (usually equivalent ones in some sense). Rewriting is at the core of theorem provers and symbolic algebra algorithms for simplification. Rewrite systems can be specification languages and even programming languages.

Recently, there has been exciting progress on efficient algorithms and decision procedures for several fundamental properties of rewrite systems including reachability, joinability, confluence, unique normalization ($UN^{\rightarrow}$), uniqueness of normal

[1] Research supported in part by the National Science Foundation grant CCF-0306475.

[2] Email:rmverma@cs.uh.edu

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

forms ($UN^{=}$) and existence of normal form for wider subclasses of rewrite systems. All these properties are well-known to be undecidable for the unrestricted case (see for instance [19]) and decidable for the ground (variable-free) case, [16,2,3,1,18,17].

Reachability, joinability and confluence were shown to be decidable for the class of linear, shallow rewrite systems [7], for right-ground systems [8], and for shallow right-linear systems [6]. $UN^{=}$ is shown decidable for linear, shallow rewrite systems [20]. $UN^{\rightarrow}$ was shown to be decidable for left-linear, right-ground systems [17] and normalization is shown to be decidable [5] (from which decidability of existence of normal forms also follows we show here) for shallow right-linear and linear right-shallow systems. Reachability and joinability are also decidable for left-linear, growing systems [14]. Progress has also been made on the termination problem but it is not studied here.

On the other hand, as far as undecidability results for these problems are concerned, latest progress can be summarized as follows. Reachability, joinability and confluence were shown undecidable for flat systems in [11] and $UN^{\rightarrow}$ is shown undecidable for linear, right shallow case in [5] and for right ground systems in [17]. $UN^{=}$ is also undecidable for right ground systems [17].

The goal of this paper is to explore the extent to which the recent decidability results can be extended to more general subclasses of rewrite systems. It turns out that for some properties - e.g., joinability, reachability, existence of normal forms and normalization - and corresponding subclasses of rewrite systems there is less room for extension and for a few (e.g., $UN^{\rightarrow}$ and $UN^{=}$) there appears to be some scope for generalization. Specifically, it is proved that the following problems are all undecidable:

(i) Reachability, joinability, existence of normal form and normalization for confluent, linear, nonoverlapping, noncollapsing, var-preserving, constructor-based, and depth two systems. These results improve the corresponding results of [19], where results were given for linear systems.

(ii) Confluence for linear, noncollapsing, constructor-based, and depth two systems.

(iii) $UN^{=}$ for linear, noncollapsing, var-preserving, and depth two systems, and for right-ground, right-flat systems.

(iv) Joinability for linear, left-flat, var-preserving and noncollapsing systems in which right-hand sides are of depth at most two.

(v) Not $UN^{\rightarrow}$ for linear, right-flat, var-preserving and noncollapsing systems in which lhs's are of depth at most two. This improves a result of [5].

(vi) Not $UN^{\rightarrow}$ for left flat, right-linear, noncollapsing systems in which right-hand sides are of depth at most two; and for right-ground, right-flat systems.

(vii) Reachability, confluence, existence of normal form, and normalization for left-flat, right-linear, noncollapsing systems in which right-hand sides are of depth at most two.

(viii) Existence of normal form and normalization for flat systems.

These undecidability results are useful since they limit the search for potential decidability results and the reachability results also yield restricted varieties of rules

that are universal for computation. Tighter reductions are also presented between some problems, the motivation being to derive new results more economically than to start from scratch every time for a decidability issue.

The organization of this paper is as follows. Section 2 defines the terms used and includes some useful new and existing results. Section 3 is devoted to two-counter machines, Section 4 to reachability, Section 5 to joinability, Section 6 to confluence, Section 7 to $UN^{\rightarrow}$ and Section 8 to existence of normal form and normalization, and Section 9 to $UN^{=}$. Section 10 concludes the paper.

2 Preliminaries

Familiarity with basic notions of rewriting is assumed [4]. Let Σ be a set, called a *signature*, with an associated *arity function* $\alpha : \Sigma \mapsto \mathbb{N}$. Let $\mathbb{V}$ be a countable set disjoint from Σ. The set $\mathbb{T}(\Sigma, \mathbb{V})$ of *terms* (over Σ) is defined as the smallest set containing $\mathbb{V}$ and such that $f(t_1, \ldots, t_n) \in \mathbb{T}(\Sigma, \mathbb{V})$ whenever $f \in \Sigma$, $\alpha(f) = n$ and $t_1, \ldots, t_n \in \mathbb{T}(\Sigma, \mathbb{V})$. The elements of the sets Σ and $\mathbb{V}$ are respectively called *function symbols* and *variables*. Note that elements a in Σ for which $\alpha(a) = 0$, called *constants*, are included in the set $\mathbb{T}(\Sigma, \mathbb{V})$. For a term s, $Var(s)$ denotes the set of variables in s. The symbols $s, t, u, \ldots$, with possible subscripts, are used to denote terms; $f, g, \ldots$, function symbols; and $a, b, \ldots$, constants. A function symbol $f \in \Sigma$ such that $\alpha(f) = m$ is also denoted by $f^{(m)}$. A term $f(t_1, \ldots, t_m)$ is written without parenthesis, $ft_1 \ldots t_m$, when it is unambiguous to do so. The *top*, $top(t)$, of a term is t if $t \in \mathbb{V}$ and is f if $t = ft_1 \ldots t_m$. The *size*, $\|t\|$, of a term is defined as 1 if $t \in \mathbb{V}$ and if $t = ft_1 \ldots t_n$ it is $1 + \sum_{i=1}^{n} \|t_i\|$. The *depth* of a term s is 0 if s is a variable or a constant, and $1 + max_i depth(s_i)$ if $s = fs_1 \ldots s_m$.

The following definitions are from [5]. A term t is called *ground* if t does not contain variables. It is called *shallow* if all variable positions in t are at depth 0 or 1. It is *flat* if its depth is at most 1. It is called *linear* if every variable occurs at most once in t.

A *position* is a possibly empty sequence of positive numbers. If p is a position and t is a term, then $t|_p$ denotes the *subterm of* t *at position* p, defined as, $t|_\lambda = t$ (where λ denotes the empty sequence) and $(ft_1 \ldots t_n)|_{i.p} = t_i|_p$ if $1 \leq i \leq n$ (and is undefined if $i > n$). Also write $t[s]_p$ (or just $t[s]$ when p is clear from the context) to denote the term obtained by replacing in t the subterm at position p by the term s. For example, if t is $f(a, g(b, h(c)), d)$, then $t|_{2.2.1} = c$, and $t[d]_{2.2} = f(a, g(b, d), d)$. A substitution, denoted by σ, is a mapping from $\mathbb{V}$ to $\mathbb{T}(\Sigma, \mathbb{V})$, homomorphically extended to a mapping from $\mathbb{T}(\Sigma, \mathbb{V})$ to $\mathbb{T}(\Sigma, \mathbb{V})$. Application of σ is denoted using a postfix notation.

An *(undirected) equation* is an unordered pair of terms, written $s = t$ satisfying the condition that both s and t cannot be distinct variables (this condition ensured nontriviality of the word problem defined below). The equation is ground if s and t are ground terms. It is var-preserving if $Var(s) = Var(t)$. A *directed equation* or *rule* is an ordered pair of terms, written $s \rightarrow t$ satisfying the condition $Var(t) \subseteq Var(s)$ and s cannot be a variable. [3] This rule is ground (shallow, linear, flat) if s and

[3] Note that these conditions are the same as in [19]. The first condition ensures that some of the problems dealt with here do not become trivial.

t are ground (shallow, linear, flat) terms. A finite set R of (ground) rules is called a *(ground) rewrite system*. It is called *right-ground* (right-shallow, right-linear, right-flat) if t is ground (shallow, linear, flat). It is called left-shallow (left-linear, left-flat) if s is shallow (linear, flat). For a rewrite system to be called shallow, left-shallow or any other such property every rule has to satisfy that property.

Terms s and t are *unifiable* if and only if there exists a ground term C which is an instance of both s and t. Say that s *overlaps* t if and only if a non-variable subterm u of one of the two terms unifies with the other term. (To check for overlaps, relabel the variables in s and t so that they do not share any variables.) A set $S \subseteq \mathcal{T}$ is *nonoverlapping* if and only if for all s, $t \in S$, *not*(s overlaps t). (Since s and t could be equal, the definition of nonoverlapping does not allow self-overlapping rules like associativity.) A system R is *nonoverlapping* if and only if the set of lhs's is nonoverlapping.

Definition 2.1 *Rewrite systems in which both left- and right-hand sides are of depth at most two will be called* depth two *systems.*

The set $D(R)$ of *defined symbols* of a rewrite system R is $D(R) = \{root(l) \mid l \rightarrow r \in R\}$. A rewrite system is called *constructor-based* if no defined symbol appears at a nonroot position in any left-hand side. A rewrite system is *var-preserving* if $Var(s) = Var(t)$ for every rule $s \rightarrow t$ in the system. The size of an equation $s = t$ or a rule $s \rightarrow t$ is defined to be $\|s\| + \|t\|$. If R is a set of rules, then we define $R^- = \{s \rightarrow t \mid t \rightarrow s \in R\}$. We say that s rewrites to t in one step at position p (by R), denoted by $s \rightarrow_{R,p} t$, if $s|_p = l\sigma$ and $t = s[r\sigma]_p$, for some $l \rightarrow r \in R$ and substitution σ. If $p = \lambda$, then the rewrite step is said to be applied *at the topmost position* (at the root) and is denoted by $s \stackrel{r}{\rightarrow}_R t$; it is denoted by $s \stackrel{nr}{\rightarrow}_R t$ otherwise. The *rewrite relation* $\rightarrow_R$ induced by R on $\mathbb{T}(\Sigma, \mathbb{V})$ is defined by $s \rightarrow_R t$ if $s \rightarrow_{R,p} t$ for some position p. The *size*, $\|R\|$, of a set R of equations or rules is the sum of the sizes of individual equations or rules in R. The cardinality of a set R is denoted by $|R|$. A rule (equation) is *collapsing* if its right-hand side (either side) is a variable. A *non-collapsing* rewrite system (equational theory) has no collapsing rules (equations).

If $\rightarrow$ is a binary relation, then $\leftarrow$ denotes its inverse, $\leftrightarrow$ its symmetric closure, $\stackrel{+}{\rightarrow}$ its transitive closure and $\stackrel{*}{\rightarrow}$ its reflexive-transitive closure. Thus, $\leftarrow_E$ and $\rightarrow_{E^-}$ denote identical relations. A *reduction sequence* of $s \stackrel{*}{\rightarrow}_R t$ (using R) is a finite sequence $s = s_0 \rightarrow_R s_1,\ s_1 \rightarrow_R s_2,\ \cdots, s_{k-1} \rightarrow_R s_k = t (k \geq 0)$, which is usually written in abbreviated form as $s = s_0 \rightarrow_R s_1 \rightarrow_R \cdots \rightarrow_R s_k = t (k \geq 0)$.

Two terms s and t are *joinable* by R, or R-joinable (notation: $s \downarrow_R t$), if there exists a term u such that $s \stackrel{*}{\rightarrow}_R u$ and $t \stackrel{*}{\rightarrow}_R u$. The terms s and t are *equivalent* by R, or R-equivalent, if $s \stackrel{*}{\leftrightarrow}_R t$ (also written $s =_R t$). A rewrite system R is *confluent* if every pair of R-equivalent terms is R-joinable. A left-linear, nonoverlapping system is also called *orthogonal* and orthogonal systems are confluent [10,15].

A term s is *irreducible* or an *R-normal form* (R will be dropped when clear from the context) if there is no term t such that $s \rightarrow_R t$. An *R-normal form of a term s* is an R-normal form t such that $s \rightarrow_R^* t$. A rewrite system is $UN^=$ (alternatively has the $UN^=$ property) if whenever $n =_R N$ for normal forms n and N implies n and N are syntactically identical. It is $UN^\rightarrow$ (alternatively has the $UN^\rightarrow$ property)

if every term has at most one R normal form.
Reachability/Joinability. *Instance:* Rewrite system, R, terms s and t. *Question:* Does $s \stackrel{*}{\rightarrow}_R t / s \downarrow_R t$?
Normal form reachability. *Instance:* Rewrite system, R, terms s and t, where t is a normal form w.r.t. R. *Question:* Does $s \stackrel{*}{\rightarrow}_R t$?

For the three problems above, also define versions in which the input rewrite system satisfies some property P. To indicate these versions, the phrase "for P systems" is attached to the basic problem.
$UN^{=}/UN^{\rightarrow}$/**Confluence**. *Instance:* Rewrite system R. *Question:* Does R have the $UN^{=}/UN^{\rightarrow}$/Confluence property?
Existence of normal form. *Instance:* Rewrite system R and term s. *Question:* Does s have an R normal form?
Word problem (WP). *Instance:* Finite set of equations E (or rewrite system R), terms s, t. *Question:* Does $s =_E t$ ($s =_R t$) ?
Normalization. *Instance:* Rewrite system R. *Question:* Does every non-variable term[4] s have an R normal form?

2.1 Some New and Existing Useful Results

In this section, we collect some existing results and observations and some new results (in particular, Corollary 2.2, Theorem 4, Theorem 5 and Corollary 2.7) that we need for the rest of the paper. As usual, $A \leq_P B$ signifies a polynomial-time reduction from A to B.

Theorem 1 ([11]) *Reachability, joinability and confluence are undecidable for flat systems.*

Theorem 2 ([5]) *Not $UN^{\rightarrow}$ is undecidable for linear right-shallow systems.*

We improve the above result of [5] below (the Appendix includes their construction to aid the reader).

Corollary 2.2 *Not $UN^{\rightarrow}$ is undecidable for linear, right-flat, var-preserving, and noncollapsing systems in which left-hand sides are of depth at most two.*

Proof. First, observe that no right-hand side of any rule in the reduction from Post correspondence problem (PCP) to $UN^{\rightarrow}$ in [5] is a variable and all right-hand sides are of depth at most one. Now there are two obstacles to proving the result: the reduction of [5] is not var-preserving, and some left-hand sides are of arbitrary depth. The first is fixed as follows. Delete constant *loop* from the signature and the rule $loop \rightarrow loop$. Next, each rule of the form $l \rightarrow loop$, which is not var-preserving and the purpose of which is to destroy undesirable normal forms, is replaced by the rule $l \rightarrow l$.

For the second, proceed as follows. Let $PCP = \{(u_i, v_i) \mid 1 \leq i \leq n\}$ be a Post Correspondence Problem over $\{a, b\}$. Replace the rule $pcp(x, y) \rightarrow pair(x, y)$ in [5] by n new rules $pcp(x, y) \rightarrow pair_{i1}(x, y)$. Now delete the rule $pair(u_i(x), v_i(y)) \rightarrow pair(x, y)$, whose left-hand sides can have arbitrary depth, in [5]. Let $m(i) = max(|u_i|, |v_i|)$. Without loss of generality assume that $|u_i| \geq |v_i|$. Then, $u_i =$

[4] Because of our restriction that lhs of a rule cannot be a variable, every variable is always a normal form.

$u_i(1), u_i(2), \ldots, u_i(m(i))$ and $v_i = v_i(1), \ldots, v_i(k(i))$ with $k(i) \leq m(i)$. Of course, $u_i(j), v_i(j) \in \{a, b\}$. There are two cases.

Case 1: $k(i) < m(i)$. Introduce $m(i)$ total rules for each i, $1 \leq i \leq n$. The first $k(i)$ of these rules are $pair_{ij}(u_i(j)(x), v_i(j)(y)) \rightarrow pair_{i(j+1)}(x, y)$, the next $m(i) - k(i) - 1$ rules are $pair_{ij}(u_i(j)(x), y) \rightarrow pair_{i(j+1)}(x, y)$ and the last rule is $pair_{im(i)}(u_i(m(i))(x), y) \rightarrow pcp(x, y)$.

Case 2: $k(i) = m(i)$. In this case the first $k(i) - 1$ rules are the same as the first $k(i) - 1$ rules in Case 1. The last rule in Case 2 is

$$pair_{im(i)}(u_i(m(i))(x), v_i(k(i))(y)) \rightarrow pcp(x, y).$$

Similarly, we handle the n rules $pair(u_i(c), v_i(c)) \rightarrow nf2$, where c is a constant.

The idea behind this construction is to remove a symbol at a time for each tile in PCP, instead of a whole string at a time as in [5]. Of course, this has to be done carefully to not mix up the symbol removal rules for the i^{th} tile in PCP with the symbol removal rules for the j^{th} tile, hence we introduce new function symbols corresponding to the tiles. We "guess" the correct tile to use with the n rules that have $pcp(x, y)$ as left-hand side and then return to the guessing stage after the tile has been "applied" to remove symbols. With this explanation it should be clear that the rest of the argument in [5] remains intact and the proof of the second part is complete. □

Theorem 3 ([12]) *Reachability is undecidable for confluent, monadic (flat right-hand sides) and semi-constructor term rewriting systems.*

In semi-constructor rewrite systems, there are restrictions on the defined symbols in the right-hand sides, but we do not make use of them here.

Notation. In all reductions below: (i) Σ denotes all the function symbols and constants in R (or E), s and t, whenever these are in the instance of the problem *being reduced*, and (ii) s and t will be assumed ground without loss of generality since otherwise we can replace each variable x by a new constant x' in these terms, consequently expanding Σ.

Theorem 4 *Joinability is undecidable for linear, left-flat, var-preserving and non-collapsing systems in which the right-hand sides are of depth at most two.*

Proof. In the reduction from PCP to Not $UN^{\rightarrow}$ [5], the authors show that two normal forms $nf1$ and $nf2$ do not have a common ancestor, which implies the system is $UN^{\rightarrow}$ since these are the only two normal forms, iff the instance of PCP has no solution. So, let R' denote the rewrite system constructed there and apply the transformation of Corollary 2.2 to get R. Now construct an instance R', s', t' of joinability as follows. $R' = R^{-}$, $s' = nf1$ and $t' = nf2$. It is easily seen that $s' \downarrow_{R'} t'$ iff R is not $UN^{\rightarrow}$ iff the instance of PCP has a solution. □

In general it can be seen that any reduction to not $UN^{\rightarrow}$ that constructs a noncollapsing, var-preserving system can be turned into a reduction to joinability since the reduction must exhibit two normal forms that have a common ancestor under certain condition and then we can use these specific normal forms in the corresponding instance of joinability. The above proof also shows that:

Corollary 2.3 *For noncollapsing, var-preserving systems: common ancestor problem $\leq_P$ joinability and joinability $\leq_P$ common ancestor problem.*

Theorem 5 *Normal form reachability and reachability are undecidable for: left-flat, right-linear, noncollapsing systems in which right-hand sides are of depth at most two.*

Proof. Let R, s, t, be an instance of joinability with R a linear, left-flat, var-preserving and noncollapsing system in which the right-hand sides are of depth at most two. This result follows from observations in [19] and Theorem 4, nevertheless it is included here to aid the reader. Construct R', s', t', an instance of reachability as follows. Let *true* be a new constant and *equal* a new binary function symbol. Let $\Sigma' = \Sigma \cup \{equal, true\}$ and $R' = R \cup \{equal(x, x) \to true\}$.

Let $s' = equal(s, t)$ and $t' = true$, which is an R' normal form. Since s and t may be assumed to be ground terms over Σ, it is easy to see that $s' \stackrel{*}{\to}_{R'} t'$ iff $s \downarrow_R t$. The reduction preserves right-linearity, shallowness, noncollapsing property and the maximum depth of the right-hand sides does not increase. Note that only one shallow, right-linear, noncollapsing rule, with right-hand side of depth 0 was added to get R'. So the rest follows from Theorem 4. □

The following three reductions (self-reducibilities) help in eliminating the non-collapsing restriction from many reductions of [19] and also simplify some of the reductions and their proofs below.

Lemma 2.4 ([17]) *WP* $\leq_P$ *WP for non-collapsing equations.*

Proof (sketch). Let E, s, t be the instance of WP. Construct E', s', t', where E' is non-collapsing as follows. Set $s' = s$ and $t' = t$. To construct E', first we discard equations of the form $x = x$ from E (if they exist) without any problem. Next, each equation of the form $l = x$ in E, where x is a variable and l not, is replaced in E' by the set of equations $\{l\sigma \to fx_1, \ldots, x_n \mid f^{(n)} \in \Sigma\}$ and $\sigma = \{x \mapsto fx_1, \ldots, x_n\}$. Here x_1,...,x_n are new variables not appearing in l. Equations of the form $x = r$, where x is a variable and r not, are handled in the same way. It is easy to verify that the reduction can be carried out in polynomial time. Correctness of the reduction is proved in the Appendix. □

Similarly, it has been proved that:

Lemma 2.5 ([17]) *Reachability* $\leq_P$ *reachability for non-collapsing systems.*

Remark 0. Note that normal form reachability reduces in polynomial time to normal form reachability for non-collapsing systems.

Lemma 2.6 ([17]) *Joinability* $\leq_P$ *joinability for non-collapsing systems.*

Note that in all three reductions above if we start with a flat system, then the depth of the left-hand sides increases by one and the right-hand sides remain flat after reduction. Therefore, by Theorem 1 and Lemmas 2.5 and 2.6 we get:

Corollary 2.7 *Reachability for non-collapsing systems, normal form reachability for non-collapsing systems and joinability for non-collapsing systems are undecidable for systems in which left-hand sides are of depth at most two and right-hand sides are flat.*

The corollary is interesting for two reasons. First, the systems constructed in [11] are collapsing systems. Hence the status of all three problems in corollary is not settled by their results. Second, it can be used to prove undecidability results

for other properties as well, which are not known to be undecidable for right-flat systems.

3 Two-Counter Machines

A two-counter machine is a Turing Machine with two semi-infinite tapes [9,13]. The tape alphabet of the machine consists of just two symbols Z and B (blank). Moreover, the symbol Z, which serves as a bottom of counter marker, appears initially in both counters. In one move the machine can change state and independently either increment, or ignore, or conditionally decrement each counter. The condition for decrementing a counter is that the counter must be positive. A transition of the two-counter machine is therefore a pair of triples of the form $((p, C_1, C_2), (q, A, B))$, where p, q are states, $C_1, C_2 \in \{Z, B\}$ and $A, B \in \{-1, 0, +1\}$, with the obvious restriction on the decrement -1 action. A two-counter machine is deterministic if no two transitions have the same first triple. As for Turing machines, the transition function of the two-counter machine is allowed to be a partial function. The configuration of a two-counter machine, and the yields in one step relation between configurations and its reflexive and transitive closure can all be defined formally and analogously to the corresponding definitions for Turing machines. Minsky proved the following theorem for two counter machines. See also [9], who show that the two counter machine can simulate a Turing machine [9].

Theorem 6 *Given a deterministic two-counter machine, the problem of determining whether the machine accepts the empty string is undecidable.*

4 Reachability

The above theorem is used to prove the following result for reachability.

Theorem 7 *Reachability is undecidable even for confluent, noncollapsing, linear, and depth-two rewrite systems. Further it remains undecidable for rewrite systems that are in addition constructor-based and var-preserving.*

Proof. The idea is to associate a rewrite system $R(M)$ with a given deterministic, two-counter machine M. Without loss of generality, we may assume that: the two-counter machine has only one final state f, it empties the two counters before accepting, and there are no transitions from the final state.

For a proof that there is such a universal two-counter machine we use the deterministic, universal, two-counter machine model of [13] in which there is only one final state q_f and there are no transitions from q_f. We add three transitions from q_f to empty the counters.

$$((q_f, B, Z), (q_f, -1, 0)), ((q_f, Z, B), (q_f, 0, -1)), ((q_f, B, B), (q_f, -1, -1))$$

and one transition $((q_f, Z, Z), (f, Z, Z))$, where f is the *only new* final state of the machine so constructed (q_f becomes a nonfinal state of the new machine).

Directly encoding the transitions of M as rules in a rewrite system is possible and seems natural but it leads to two problems: overlapping rules and the depth of variables can exceed two. To avoid these problems two mechanisms are used: a

checking module and a "delayed" two-step incrementing procedure for incrementing nonempty counters.

There is a constant p in the signature of the rewrite system for every state p of the two-counter machine and a constant Z to represent the empty counter. The blank symbol is simulated by a unary symbol B. The signature also contains a ternary symbol h, a binary symbol $equal$, several auxiliary function symbols, and the constant $true$.

First, we group all the transitions of M that take place on the same state, i.e., for which the first component of the first triple in the transition is the same, together. If the common state for a transition group is p, we call them p-transitions. For each p-transition group of M we have a rule in R of the form $h(p, x, y) \rightarrow h_p(equal(x, Z), equal(y, Z))$. The rules for equal are two: $equal(Z, Z) \rightarrow true$ and $equal(B(x), Z) \rightarrow B(x)$. Next, we have up to four rules of the following forms, where the right-hand sides are generic terms to be specified below:

(i) $h_p(true, true) \rightarrow h(q, A_1, B_1)$

(ii) If M increments the second counter, then the rule is $h_p(true, B(x)) \rightarrow h_{p2}(q, A_2, B_2)$. If it does not, then the rule is $h_p(true, B(x)) \rightarrow h(q, A_2, B_2)$.

(iii) If M increments the first counter, then the rule is $h_p(B(x), true) \rightarrow h_{p1}(q, A_3, B_3)$ otherwise it is $h_p(B(x), true) \rightarrow h(q, A_3, B_3)$.

(iv) If M increments both counters, the rule is

$$h_p(B(x), B(y)) \rightarrow h_{p12}(q, A_4, B_4).$$

If it increments the first only, the rule is $h_p(B(x), B(y)) \rightarrow h_{p1}(q, A_4, B_4)$. If it increments the second only, the rule is $h_p(B(x), B(y)) \rightarrow h_{p2}(q, A_4, B_4)$. If it does not increment either counter, the rule is $h_p(B(x), B(y)) \rightarrow h(q, A_4, B_4)$.

When M increments a non-empty counter, the simulation needs to break this into two reduction steps to avoid increasing the depth of variables beyond two. The right-hand side of these rules are obtained from the right-hand sides of the p-transitions by finding the transitions that apply to the four cases: both counters are initially zero; the first counter is initially zero, the second is not; the first is not zero, the second is; and both counters are non-zero initially. The constant q is determined by the state component of the second triple of the corresponding transition. A_1 (also B_1, A_2, B_3) is Z if M ignores the counter and $B(Z)$ if it increments it (it cannot decrement any counter in this case since the counter is zero). B_2 (A_3, A_4) is x if M decrements the corresponding counter, and $B(x)$ if it ignores or increments the corresponding counter. B_4 is y if M decrements the corresponding counter and $B(y)$ if it ignores or increments it.

The rules for incrementing the correct counters are introduced as needed and are as follows:

$$h_{p1}(q, x, y) \rightarrow h(q, B(x), y)$$
$$h_{p2}(q, x, y) \rightarrow h(q, x, B(y))$$
$$h_{p12}(q, x, y) \rightarrow h(q, B(x), B(y))$$

The specification of $R(M)$ is complete. By inspection, $R(M)$ is linear, var-preserving, constructor-based and depth-two. It is also nonoverlapping since M

is deterministic and by the construction. Since it is linear and nonoverlapping, it is also confluent.

Now let s be the term $h(s_0, Z, Z)$, where s_0 is the initial state of M, and let t be the term $h(f, Z, Z)$ where f is the only final state of the two-counter machine. We prove that $s \xrightarrow{*}_{R(M)} t$ iff the two-counter machine M accepts the empty string. The proof of the if direction is straightforward by induction on the number of steps taken by M in the accepting computation.

For the only if direction, we prove the following more general result: if $h(p, A_1, A_2) \xrightarrow{*}_{R(M)} h(q, B_1, B_2)$, where p, q are any state constants, and $A_i, B_i \in B^*(Z)$ [5] for $i \in \{1, 2\}$, then the configuration (p, A_1, A_2) yields in 0 or more steps of M the configuration (q, B_1, B_2). Clearly, this result implies what is needed above. The proof of the more general result proceeds by induction on the number of intermediate terms of the form $h(\ldots)$ in the reduction sequence. Details can be easily filled in by the reader. This completes the proof. □

5 Joinability

The proof of Theorem 7 also yields the following results:

Theorem 8 (i) *Joinability is undecidable for confluent, linear, nonoverlapping, noncollapsing, var-preserving, constructor-based, and depth-two rewrite systems.*

(ii) *Normal form reachability is undecidable for confluent, linear, nonoverlapping, noncollapsing, var-preserving, constructor-based, and depth-two rewrite systems.*

Proof. Because we started with a two-counter machine that has no transitions from the final state f, the term $h(f, Z, Z)$ is a normal form. This immediately proves the second statement. Now $h(s_0, Z, Z) \downarrow_{R(M)} h(f, Z, Z)$ iff $h(s_0, Z, Z) \xrightarrow{*}_{R(M)} h(f, Z, Z)$ since $h(f, Z, Z)$ is a normal form. Thus, the first statement also follows. □

6 Confluence

The following reduction is from [17]. A proof sketch is included to show how the depth bound changes in the reduction. It is partly recovered in the Remark afterward.

Theorem 9 ([17]) *Normal form reachability for non-collapsing rewrite systems $\leq_P$ confluence.*

Proof (sketch). Let R, s, t be an instance of normal form reachability, where t is an R-normal form and R is non-collapsing. Let a be a new constant and h a new binary function symbol not in Σ. Let $\Sigma' = \Sigma \cup \{h, a\}$.

Let $R_1 = \{fx_1, \ldots, x_n \to h(s, fx_1, \ldots x_n) \mid f^{(n)} \in \Sigma' - \{a\}\}$.

Let $R' = R \cup R_1 \cup \{h(a, x) \to a\} \cup \{t \to a\}$.

[5] 0 or more applications of B

Note that every term u in $\mathbb{T}(\Sigma', \mathbb{V})$, except variables and a, reduces via the R_1 rules to the term $h(s, u)$. Since R is non-collapsing and the new rules are non-collapsing, R' is non-collapsing. Moreover, since no left-hand side of a rule in R can be a variable and the new rules also satisfy this condition, so R' also satisfies this condition. It is shown in [17] that R' is confluent iff $s \stackrel{*}{\rightarrow}_R t$. □

Remark 1. The above reduction does not preserve groundness, but it does preserve left-linearity, right-linearity, linearity, and noncollapsing property. It also preserves left-flatness and the right-hand sides of the new rules introduced are of depth at most two (for this s and t must be flattened first - using the procedure of [7] for example - and their new names must be used in the new rules).

Corollary 6.1 *Normal form reachability for non-collapsing rewrite systems $\leq_P$ local confluence.*

Proof. A very similar argument to that given in the proof of Theorem 9. Note that all critical pairs in R' are joinable iff $s \stackrel{*}{\rightarrow}_R t$. □

Corollary 6.2 *Local confluence and confluence are undecidable for:*

(i) left-flat, right-linear and noncollapsing systems in which right-hand sides are of depth at most two.

(ii) linear, noncollapsing, and depth two systems.

Proof. (i) Follows from Theorem 5 and Theorem 9. (ii) Follows from Theorem 7 and Theorem 9. □

7 Unique Normalization

We now prove several limits on $UN^{\rightarrow}$: for left-flat, right-linear, noncollapsing systems in which right-hand sides are of depth at most two; for linear, noncollapsing and depth two systems; and for right-ground, right-flat systems.

Observe that the reduction from joinability in [17] to not $UN^{\rightarrow}$ cannot be used here since that reduction does not preserve flatness and left-linearity. We wish to preserve flatness and linearity so we need a more complex reduction and proof and we use reachability as the starting point.

Note that in a non-collapsing system it is not possible to reduce a non-variable term to a variable. Hence we cannot have violation of $UN^{\rightarrow}$ involving a normal form that is a variable.

Theorem 10 *Reachability for non-collapsing rewrite systems $\leq_P$ not $UN^{\rightarrow}$.*

Proof. Let R, s, t be an instance of reachability with R non-collapsing. We first flatten s and t using flattening rules (see for example [7]). These rules preserve R reachability of terms and are flat. Let c_s and c_t be the new "names" for s and t respectively. We add all the new names so introduced to Σ and the flattening rules to R. Let $true, false$ be two new constants and $H = \{h_f \mid f^{(m)} \in \Sigma, m > 0\}$ be a set of new function symbols not in Σ. For each $f^{(m)} \in \Sigma$ the corresponding function symbol h_f has arity $m + 2$. Let $\Sigma' = \Sigma \cup H \cup \{true, false\}$.

Let $T = \{c \rightarrow c \mid c \in \Sigma' - \{true, false\}\} \cup \{fx_1 \ldots x_n \rightarrow fx_1 \ldots x_n \mid f^{(n)} \in \Sigma', n > 0\}$. So that all non-variable terms in $\mathbb{T}(\Sigma', \mathbb{V})$ are reducible except $true$ and $false$. Let

$S = \{c \to h(c_s, c) \mid c \in \Sigma' - \{true, false\}\} \cup \{fx_1 \ldots x_n \to h_f(c_s, x_1 \ldots x_n) \mid f^{(n)} \in \Sigma\}$.

Let $V = \{h_f(c_t, y_1, \ldots, y_n) \to true \mid f^{(n)} \in \Sigma\}$.

Let $R' = R \cup S \cup T \cup V \cup \{c_t \to false\}$.

To show that R' is not $UN^{\to}$ iff $s \stackrel{*}{\to}_R t$. Suppose that R' is not $UN^{\to}$. Consider, a term $A \in \mathbb{T}(\Sigma', \mathbb{V})$ and suppose that A has two distinct normal-forms B and C. Because of the non-collapsing requirement on R and construction of R', R' is also non-collapsing and so neither B nor C can be a variable. Hence, we must have reduction sequences $p\colon A \stackrel{*}{\to}_{R'} true$ and $q\colon A \stackrel{*}{\to}_{R'} false$. We claim that p and q imply $s \stackrel{*}{\to}_R t$. Note that since R' is a rewrite system A cannot be a variable. Then, $top(A) \in \Sigma$, since otherwise $A \stackrel{*}{\to}_{R'} false$ is not possible. Now since $top(A) \in \Sigma$ and there are no collapsing rules, the rule $top(A)(x_1, \ldots, x_n) \to h_f(c_s, x_1, \ldots, x_n)$ must have been applied at the root in sequence p and subsequently the rule $h_f(c_t, y_1, \ldots, y_n) \to true$ must have been applied at the root in p. This implies that $c_s \stackrel{*}{\to}_{R'} c_t$, which implies $s \downarrow_{R'} t$. Now, through a similar argument as in the full proof of Theorem 9 ([17]) we get that $s \stackrel{*}{\to}_R t$. Now suppose that $s \stackrel{*}{\to}_R t$, which implies $s \stackrel{*}{\to}_{R'} t$. Clearly, R' is not $UN^{\to}$ since $t \stackrel{*}{\to}_{R'} c_t \to_{R'} h(c_s, c_t) \stackrel{*}{\to}_{R'} h(c_t, c_t) \to_{R'} true$ and $t \stackrel{*}{\to}_{R'} c_t \to_{R'} false$ and $true$ and $false$ are distinct normal forms. □

Note that the above reduction preserves flatness and linearity.

Corollary 7.1 *Not $UN^{\to}$ is undecidable for:*

(i) left-flat, right-linear, noncollapsing systems in which right-hand sides are of depth at most two.

(ii) linear, noncollapsing, and depth-two systems.

Proof. (i) Theorem 10 and Theorem 5 (ii) Theorem 10 and Theorem 7. □

In [17], it is shown that $UN^{\to}$ is undecidable for right-ground systems. We can apply the flattening procedure of [20], which preserves the $UN^{\to}$ property to get:

Theorem 11 *$UN^{\to}$ is undecidable for right-ground, right-flat rewrite systems.*

8 Existence of Normal Form and Normalization

Theorem 12 *Existence of normal form is undecidable for confluent, linear, non-collapsing, var-preserving, constructor-based and depth two systems.*

Proof. Let Σ denote the signature of $R(M)$ constructed in Theorem 7 above. We construct R' and s', an instance of existence of normal form, as follows. Let $T = \{c \to c \mid c \in \Sigma\} \cup \{hx_1, \ldots, x_n \to hx_1, \ldots, x_n \mid h \in \Sigma\}$, so that all non-variable terms in $\mathbb{T}(\Sigma, \mathbb{V})$ are reducible.

Let $R' = R(M) \cup T \cup \{h(f, Z, Z) \to d\}$, where d is a new constant, and let $s' = h(s_0, Z, Z)$. Then, clearly s' has an R' normal form (viz. d) iff $s \stackrel{*}{\to}_R h(f, Z, Z)$. Note that R' is var-preserving, non-collapsing, linear, and depth two. It is also confluent since all critical pairs are parallel closed [10] trivially. □

Theorem 13 *Reachability $\leq_P$ existence of normal form.*

Proof. Let R, s, t be an instance of reachability. Construct R' and s', an instance of

existence of normal form, as follows. First flatten t using flattening rules [7]. These rules preserve R reachability of terms. Let c_t be the new "name" for t. We add all the new names so introduced to Σ and the flattening rules to R. Let d be a new constant not in Σ. Let $T = \{c \to c \mid c \in \Sigma\} \cup \{hx_1, \ldots, x_n \to hx_1, \ldots, x_n \mid h \in \Sigma\}$.

Let $R' = R \cup T \cup \{c_t \to d\}$ and $s' = s$.

Then, clearly s' has an R' normal form (viz. d) iff $s \stackrel{*}{\to}_R t$. Recall that s can be assumed ground. We do not assume a fixed signature so this does not affect the normal form property of s since the new constants are R-irreducible (because they are new). If we have to introduce new constants to force groundness of s, t, then we make them part of Σ and make them reducible w.r.t. R'. □

The above reduction preserves left-linearity, right-linearity, linearity, flatness, noncollapsing property and is var-preserving. It can also be modified to preserve groundness (see [19]).

Corollary 8.1 *(a) Existence of normal form is undecidable for: (i) flat systems and (ii) left-flat, right-linear, noncollapsing, var-preserving systems in which right-hand sides are of depth at most two.*

(b) Normalization is undecidable for: (i) linear, noncollapsing, var-preserving and depth two systems, (ii) flat systems, and (iii) left-flat, right-linear, noncollapsing, var-preserving systems in which right-hand sides are of depth at most two.

(c) Existence of normal form is decidable for: (i) linear, right-shallow and (ii) shallow, right-linear systems.

Proof. (a) (i) Theorem 13 and Theorem 1. (ii) Theorems 13 and 5.

(b) Reachability can be reduced to normalization. The proof is quite similar to that of the reduction to existence of normal forms problem so we describe only the changes. Using the notations of Theorem 13 construct R'' an instance of normalization as follows: $R'' = R' \cup S$, where R' is as in Theorem 13 and $S = \{c \to c_s \mid c \in \Sigma\}$ $\cup \{hx_1, \ldots, x_n \to c_s \mid h^{(n)} \in \Sigma, n > 0\}$. Recall that c_s is obtained by flattening s. If s is flat, we still introduce the constant c_s and the auxiliary rules. However, we will show something stronger, viz., the existence of a reduction between existence of normal form and normalization.

Existence of normal form is a special case of the normalization problem. In case that is not convincing to the reader, we show that there exists a reduction from existence of normal form to the normalization problem as follows. Let R, s be an instance of the existence of normal form problem. Flatten s as in [20] and call the resulting rewrite system also R. Construct a Turing Machine N that accepts exactly the ground R normal forms in a single final state, say f. Now simulate N using a two-counter machine say M. Next simulate M using a rewrite system $R(M)$ and let $\Sigma(M)$ denote the alphabet of $R(M)$ (we ensure that f is a constant in $\Sigma(M)$).

Now using the notations of Theorem 13 construct R' an instance of normalization as follows: $R' = R \cup S \cup R(M)$, where $S = \{c \to c_s \mid c \in \Sigma \cup \Sigma(M) - \{f\}\} \cup$ $\{hx_1, \ldots, x_n \to c_s \mid h^{(n)} \in \Sigma, n > 0\}$. The S rules ensure that f is the only ground normal form of R' and that every non-variable term, except f, reduces to c_s, the new name for s. Then, every term has an R' normal form iff s has an R normal form. Notice the reduction preserves linearity and depth two. It can be modified to preserve flatness as follows. Instead of simulating N by a two-counter machine

construct a PCP instance and then simulate the PCP instance via a rewrite system as in [11]. Details of this construction are deferred to the full version of this paper.

(c) Both parts follow from the reduction of part (b) to normalization and decidability results for normalization in [5]. □

9 Uniqueness of Normal Forms

As a corollary of Theorem 8 and the reduction from joinability for confluent systems to uniqueness of normal forms [19], we have the following result:

Theorem 14 *$UN^=$ is undecidable for linear, noncollapsing, var-preserving, and depth-two rewrite systems.*

Proof. The reduction of [19] preserves linearity, noncollapsing, and var-preserving properties and can be modified using the flattening procedure of [20] on the (ground) terms s and t to preserve the depth. □

In [17], it is shown that $UN^=$ is undecidable for right-ground systems. We can apply the flattening procedure of [20], which preserves the $UN^=$ property to get:

Theorem 15 *$UN^=$ is undecidable for right-ground, right-flat rewrite systems.*

10 Conclusions and Future Work

This paper shows that for several fundamental properties of rewrite systems the class of linear, depth two systems is a close boundary as far as restrictions on occurrences of variables and their depth is concerned. Further, joinability is undecidable for linear, left-flat systems in which the right-hand sides are of depth at most two. This is a sharper lower bound than linear, depth two systems. Reachability, confluence and unique normalization, existence of normal form and normalization are all undecidable for left-flat, right-linear systems in which right-hand sides are of depth at most two as well. This represents an exchange of left-linearity with left-flatness. Existence of normal form and normalization are also undecidable for flat systems. Thus, these results together with [19], [14], [20], and [5] leave the following four out of 15 subclasses [6] of systems as far as the last two properties and reachability are concerned: left-linear, right-shallow; left-linear, left-shallow; right-linear, right-shallow; and linear, left-shallow for which the status of these two properties is still open as far as I know. The status of confluence is open for two more subclasses, viz., linear, right-shallow systems and left-linear, shallow systems to my knowledge. The status of joinability is now open for only three out of 15 subclasses. The situation for $UN^{\rightarrow}$ and $UN^=$ is the least satisfactory, which are open for seven and 10 subclasses respectively to my knowledge. Of course, this classification is with respect to occurrences of variables and depth of variables, which is the scope of this paper. Other properties may be brought to bear (e.g., see [12]) opening new vistas.

[6] 1. shallow 2. linear 3. left-linear, right-shallow 4. left-shallow, right-linear 5. left-linear, shallow 6. left-shallow, linear 7. right-linear, shallow 8. linear, right-shallow 9. linear, shallow. 10. left-linear, left-shallow 11. right-linear, right-shallow. 12. left-linear 13. left-shallow 14. right-linear 15. right-shallow

References

[1] H. Comon, G. Godoy, and R. Nieuwenhuis. Confluence of ground rewrite systems is in P. In *Proc. IEEE Symp. on Foundations of Computer Science*, 2001.

[2] M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewrite systems. *Information and Computation*, 88:187–201, 1990. Also in Proc. IEEE Symp. on LICS 1987.

[3] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. IEEE Conf. on Logic in Computer Science*, pages 242–248, 1990.

[4] N. Dershowitz and D. Plaisted. Rewriting. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 9, pages 535–610. Elsevier Science, 2001.

[5] G. Godoy and S. Tison. On the normalization and unique normalization properties of term rewrite systems. In *Proc. Conf. on Automated Deduction*, pages 247–262, 2007.

[6] G. Godoy and A. Tiwari. Confluence of shallow right-linear rewrite systems. In *CSL*, pages 541–556, 2005.

[7] G. Godoy, A. Tiwari, and R. Verma. On the confluence of linear shallow rewrite systems. *Proceedings of the Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science*, 2003.

[8] G. Godoy, A. Tiwari, and R. Verma. Characterizing confluence by rewrite closure and right ground term rewrite systems. *Applicable Algebra in Engineering, Communication and Computing*, 15(1):13 – 36, 2004.

[9] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Co., 1979.

[10] G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980. Also in 18th IEEE FOCS, 1977.

[11] I. Mitsuhashi, M. Oyamaguchi, and F. Jacquemard. The confluence problem for flat TRSs. In *8th Artificial Intelligence and Symbolic Computation Conference*, pages 68–81, 2006.

[12] I. Mitsuhashi, M. Oyamaguchi, and T. Yamada. The reachability and related decision problems for monadic and semi-constructor TRSs. *Inf. Process. Lett.*, 98(6):219–224, 2006.

[13] K. Morita. Universality of a reversible two-counter machine. *Theoretical Computer Science*, 168(2):303–320, 1996.

[14] T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *Inf. Comput.*, 178(2):499–514, 2002.

[15] M. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *Lecture Notes in Computer Science*. Springer-Verlag, 1977.

[16] M. Oyamaguchi. The church rosser property for ground term rewriting systems is decidable. *Theoretical Computer Science*, 49:43–79, 1987.

[17] R. Verma. Complexity of normal form properties and reductions for rewriting problems. *Fundamenta Informaticae*. accepted for publication.

[18] R. Verma and A. Hayrapetyan. A new decidability technique for ground term rewriting systems with applications. *ACM Transactions on Computational Logic*, 6(1):102–123, 2005.

[19] R. Verma, M. Rusinowitch, and D. Lugiez. Algorithms and reductions for rewriting problems. *Fundamenta Informaticae*, 46(3):257–276, 2001. Also in Proc. of Int'l Conf. on Rewriting Techniques and Applications 1998.

[20] J. Zinn and R. Verma. A polynomial-time algorithm for uniqueness of normal forms of linear, shallow rewrite systems. In *Proc. IEEE Conf. on Logic in Computer Science*, 2006. short presentation.

11 Appendix

Lemma 11.1 ([17]) *WP* $\leq_P$ *WP for non-collapsing equations.*

Proof. Let E, s, t be the instance of WP. Construct E', s', t', where E' is non-collapsing as follows. Set $s' = s$ and $t' = t$. To construct E', first we discard equations of the form $x = x$ from E (if they exist) without any problem. Next, each

equation of the form $l = x$ in E, where x is a variable and l not, is replaced in E' by the set of equations $\{l\sigma \to fx_1, \ldots, x_n \mid f^{(n)} \in \Sigma\}$ and $\sigma = \{x \mapsto fx_1, \ldots, x_n\}$. Here x_1,...,x_n are new variables not appearing in l. Equations of the form $x = r$, where x is a variable and r not, are handled in the same way. It is easy to verify that the reduction can be carried out in polynomial time.

The correctness of the reduction now follows from the following proposition and the fact that we may assume s and t are ground.

Proposition 11.2 *Given E, s and t, with s and t ground, if $s =_E t$, then there is an equational proof $s =_E t$ such that every equation instance used in this proof contains only function symbols and constants from Σ. Moreover, if s and t are ground, then there exists an equational proof with the additional property that every equation instance used in this proof is ground.*

Proof. By induction on the length of the proof $s =_E t$ using the fact that any new symbols must be in the substitution part of the equation used and so may be replaced with symbols that appear in terms of E, s or t. For the groundness part, observe that we can define a substitution σ that maps variable x to any constant in Σ and by stability of equational proofs under substitutions we get that $s\sigma =_E t\sigma$, but since s and t are ground so $s = s\sigma =_E t\sigma = t$. □

11.1 The Godoy and Tison [5] Construction

To make the paper self-contained we include the reduction of PCP to $UN^{\to}$ given in [5]. Let PCP $= \{(u_i, v_i) \mid 1 \leq i \leq n\}$ be a Post Correspondence Problem over $\{a, b\}$. The TRS defined by [5] is: $pcp(x, y) \to eq(x, y)$, $pcp(x, y) \to pair(x, y)$, $eq(a(x), a(y)) \to eq(x, y)$, $eq(b(x), b(y)) \to eq(x, y)$, $eq(c, c) \to nf1$, $pair(u_i(x), v_i(y)) \to pair(x, y), 1 \leq i \leq n$, $pair(u_i(c), v_i(c)) \to nf2, 1 \leq i \leq n$, $a(x) \to loop$, $b(x) \to loop$, $c \to loop$, $pair(x, y) \to loop$, $eq(x, y) \to loop$, $pcp(x, y) \to loop$, and $loop \to loop$.