The Confluence of Ground Term Rewrite Systems is Decidable in Polynomial Time*

Hubert Comon
Stanford University and ENS Cachan
61 Ave. Pr. Wilson,
F-94235 Cachan cedex, France.
comon@lsv.ens-cachan.fr

Guillem Godoy
Technical University of Catalonia,
Jordi Girona 1, 08034 Barcelona, Spain.
ggodoy@lsi.upc.es

Robert Nieuwenhuis
Technical University of Catalonia,
Jordi Girona 1, 08034 Barcelona, Spain.
roberto@lsi.upc.es

Abstract

The confluence property of ground (i.e., variable-free) term rewrite systems (GTRS) is well-known to be decidable. This was proved independently in [4, 3] and in [13] using tree automata techniques and ground tree transducer techniques (originated from this problem), yielding EXPTIME decision procedures (PSPACE for strings). Since then, it has been a well-known longstanding open question whether this bound is optimal (see, e.g., [15]).

Here we give a polynomial-time algorithm for deciding the confluence of GTRS, and hence as well for the particular case of suffix- and prefix string rewrite systems or Thue systems. We show that this bound is optimal for all these problems by proving PTIME-hardness for the string case. This result may have some impact as well on other areas of formal language theory and, in particular, on the theory of tree automata.

1. Introduction

The fields of rewriting theory, formal language theory (and in particular automata theory), and algebra have provided many useful tools to each other [1, 12]. The theory of rewriting also contributes with fundamental results to programming languages (semantics, implementation, static analysis) and automated deduction (symbolic computation, decision procedures, combination procedures, constraint solving, parallel deduction), see, e.g., [5, 6] and their references.

Rewriting essentially consists of the application of rules to expressions, replacing subexpressions by other ones (usually equivalent ones, in some sense).

Example 1 *A (bottom up)* tree automaton *essentially consists of a ground term rewrite system, like*

$$\begin{array}{rcl} a & \to & q_a \\ g(q_a) & \to & q_g \\ f(q_g, q_f) & \to & q_{accept} \end{array} \qquad \begin{array}{rcl} g(q_g) & \to & q_g \\ f(q_a, q_a) & \to & q_f \end{array}$$

for an automaton recognizing the regular tree language $f(g^+(a), f(a, a))$.

String rewrite systems or Thue systems are a particular case of term rewrite systems, restricted to the language of unary function symbols and constants. A fundamental property of term rewrite systems is the *confluence* or *Church-Rosser* property: it essentially says that if some expression is reached by successive rewrite (or *reduction*) steps, then this is also the case independently of which subexpression is rewritten first and by which rule, i.e., confluence converts nondeterminism from "don't know" into "don't care", thus avoiding the need for backtracking. Therefore confluence ensures the uniqueness of expressions in normal (i.e., irreducible) form. For instance, the tree automaton of Example 1 can be regarded as deterministic since the rewrite system is confluent.

Confluence is of course a well-studied property for the different classes of rewrite systems. It is undecidable in general (see e.g., [8] and its references), it is decidable for

*Second and third authors partially supported by the Spanish CICYT project HEMOSS ref. TIC98-0949-C02-01; second author supported by Departament d'Universitats, Recerca i Societat de la Informació de la Generalitat de Catalunya.

terminating systems [9], and, interestingly, it is also decidable for arbitrary ground systems. This latter result was proved independently in [4, 3] and in [13] using tree automata techniques and ground tree transducer techniques (originated from this problem), yielding in both cases EXPTIME decision procedures. The basic idea is that one can build a polynomial-size tree automaton A_1 accepting the triples $\langle t, u, v\rangle$ such that u and v are reachable from t, and another one A_2 accepting the triples $\langle t, u, v\rangle$ such that u and v are joinable. Then, roughly, confluence amounts to the inclusion $A_1 \subseteq A_2$, which is in EXPTIME.

Since these ideas were developed, it has been a well-known open question whether this bound is optimal (see, e.g., [15]), but up to now, no algorithms better than PSPACE and EXPTIME had been found for ground string and term rewrite systems, respectively, and no hardness results for these complexity classes were found either.

In this paper we give the first polynomial time decision procedure for the confluence of arbitrary ground term rewrite systems (GTRS). Note that GTRS properly include suffix string rewrite (or Thue) systems (or, equivalently, prefix ones). Confluence of GTRS contains many interesting problems as instances. For example, there is a simple reduction for uniform word problems in finitely presented algebras: replace axioms $u = v$ with two rules $u \to v$ and $v \to u$ and, if the problem is to decide $s = t$, add two rules $0 \to s$, $0 \to t$, where 0 is a new constant. Then $s = t$ is a consequence of the axioms if, and only if, the resulting system is confluent. D. Kozen in [10] shows that the uniform word problems for ground equations (i.e. without variables) is PTIME-complete (see also [7, 11, 17]). Our result subsumes this one, and completes the picture for the ground rewriting case; all properties of interest are decidable in polynomial time for this class: reachability, joinability [3, 13], termination [14], word problem [10] and confluence. This class of rewriting systems is the only known class with tractability results. In contrast, there are plenty of undecidability results for finitely presented monoids [12].

The key ideas for our algorithm are as follows. In Section 3 we apply three polynomial-time confluence-preserving transformations to the GTRS. First, the GTRS is converted into Curry form, which is well-known from the implementation of functional languages. In Curry form, all terms are built over a single binary *apply* function symbol and constants. Second, by introducing more new constants, equated by bi-directional rules to *deep* terms, we can limit the depth of all sides of rules to two (note that the rewrite rules of tree automata like the one of Example 1 also have depth at most two). Similar transformations are used in the well-known fast algorithms for congruence closure of [7, 11, 17]. Third, we rely on some techniques introduced by Plaisted in his polynomial time decision procedure for termination of GTRS [14]: we transform the GTRS in polynomial time into another GTRS by which all terms reachable from (subterms of) sides of rules can also be reached by *increasing* rewrite derivations: derivations where no step takes place at a position at or above the position of some previous step. For example, with the rules $a \to b$ (1) and $f(b, b) \to f(a, a)$ (2), the derivation $\underline{f(b, b)} \to_2 f(a, \underline{a}) \to_1 f(a, b)$ is increasing whereas $f(b, \underline{a}) \to_1 \underline{f(b, b)} \to_2 f(a, a)$ is not.

In Section 4 we develop another essential ingredient for our result. We give a polynomial time analysis of what we have called *stable root symbols*. In a term $f(s, t)$, the root symbol f is stable if all terms reachable from it are rooted by f. Then we say that the root symbol of $f(s, t)$ can be *stabilized* by R if f is stable in $f(s', t')$ for some s' and t' reachable from s and t respectively. We show that this latter property is decidable in polynomial time.

After the transformations, and using the root stabilization test, in Section 5 several simple, efficiently checkable, necessary conditions for confluence of R are defined. These conditions can hence be assumed to hold in subsequent steps of our algorithm, which considerably simplifies matters.

Section 6 presents the main algorithm. It is based on the stability analysis and on a notion which we have called *deep joinability*: two terms s and t are called deeply joinable if s' and t' are joinable (i.e., they rewrite into the same term) whenever $s \to_R^* s'$ and $t \to_R^* t'$. A polynomial-time fixpoint computation allows us to compute the deep joinability relation for all pairs of subterms of sides of R. This gives us our main result, since R is confluent if, and only if, all left hand sides of rules of R are deeply joinable with themselves.

In Section 7 we show that our results are optimal by proving PTIME-hardness for the string case. Finally, in Section 8 concluding remarks are given, and more precise complexity issues are discussed. We also mention some possible implications to automata theory, and some other open problems for more general classes of term rewrite systems that could be attacked using the techniques introduced here.

2. Basic Notions

Let $\mathcal{F}$ be a (finite) set of function symbols with an arity function *arity*: $\mathcal{F} \to I\!N$. Function symbols f with *arity*$(f) = n$ are called n-*ary* symbols (when $n = 1$, one says *unary* and when $n = 2$, *binary*). If *arity*$(f) = 0$, then f is a *constant symbol*. The set of ground terms over $\mathcal{F}$, denoted by $\mathcal{T}(\mathcal{F})$, is the smallest set containing all constant symbols such that $f(t_1, \ldots, t_n)$ is in $\mathcal{T}(\mathcal{F})$ whenever $f \in \mathcal{F}$, *arity*$(f) = n$, and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$. A *position* is a sequence of positive integers. If p is a position and t is a term, then by $t|_p$ we denote the *subterm of* t *at* *position* p: we have $t|_\lambda = t$ (where λ denotes the empty sequence) and $f(t_1, \ldots, t_n)|_{i.p} = t_i|_p$ if $1 \leq i \leq n$ (and is undefined if $i > n$). We also write $t[s]_p$ 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 $t[d]_{2.2} = f(a, g(b, d), d)$. By $|s|$ we denote the *size* (number of symbols) of a term s: we have $|a| = 1$ if a is a constant symbol and $|f(t_1, \ldots, t_n)| = 1 + |t_1| + \ldots + |t_n|$. The *height* (or *depth*) of a term s is denoted by $height(s)$ and is defined: $height(a) = 1$ if a is a constant symbol and $height(f(t_1, \ldots, t_n)) = 1 + max(height(t_1), \ldots, height(t_n))$.

If $\rightarrow$ is a binary relation on a set S, then $\rightarrow^+$ is its transitive closure and $\rightarrow^*$ is its reflexive-transitive closure. The relation $\rightarrow$ is *confluent* or *Church-Rosser* if the relation $\leftarrow^* \circ \rightarrow^*$ is contained in $\rightarrow^* \circ \leftarrow^*$, that is, for all s, t_1 and t_2 in S such that $s \rightarrow^* t_1$ and $s \rightarrow^* t_2$, there exists an s' in S such that $t_1 \rightarrow^* s'$ and $t_2 \rightarrow^* s'$.

A *ground rewrite rule* is pair of ground terms (l, r), denoted by $l \rightarrow r$; then l and r are the left- and right hand sides of $l \rightarrow r$, respectively. A *ground term rewrite system* (GTRS) R is a finite set of ground rewrite rules. The rewrite relation with R on $\mathcal{T}(\mathcal{F})$ is the smallest relation $\rightarrow_R$ such that $s[l]_p \rightarrow_R s[r]_p$ whenever $l \rightarrow r \in R$. A GTRS R is said to be confluent if the relation $\rightarrow_R$ is confluent. The *size* $|R|$ of a GTRS R is the sum of $|l| + |r|$ for $l \rightarrow r \in R$.

3. Transformations on the GTRS

3.1. Transformation into Curry terms

Consider a new signature $\mathcal{F}_c$ obtained from the original $\mathcal{F}$ by introducing a new binary function symbol f, and converting all other symbols into constants. Then the *Curry form* (well-known from functional programming) of a term t in $\mathcal{T}(\mathcal{F})$ is a term $Curry(t)$ in $\mathcal{T}(\mathcal{F}_c)$ defined as follows:

$Curry(c) = c$, if c is a constant symbol, and

$Curry(g(t_1 ... t_n)) =$

$f(\ldots f(\ f(g, Curry(t_1)), Curry(t_2)), \ldots, Curry(t_n))$

For example, the Curry form of $g(a, h(b), c))$ is $f(f(f(g, a), f(h, b)), c)$. Similarly, we consider the $Curry$ transformation on rewrite rules and GTRS. Then we have the following result.

Lemma 2 $Curry(s) \rightarrow_{Curry(R)} Curry(t)$ *if, and only if,* $s \rightarrow_R t$.
Moreover, R *is confluent on* $\mathcal{T}(\mathcal{F})$ *if, and only if,* $Curry(R)$ *is confluent on* $\mathcal{T}(\mathcal{F}_c)$.

Since this transformation is polynomial, in what follows w.l.o.g. we assume R to be built over a signature $\mathcal{F}$ with only the binary symbol f and constants.

3.2. Transformation into terms of depth at most 2

Consider the following transformation step on R:

$$R \Rightarrow R' \cup \{c \rightarrow t, t \rightarrow c\} \qquad (\textit{Constant introduction and replacement})$$

where c is a new constant symbol not occurring anywhere in R, and R' is obtained by replacing all occurrences of t in R by c. Similar transformations are used as well in fast algorithms for congruence closure [7, 11, 17]. We have the following result.

Lemma 3 *If* $R_1 \Rightarrow R_2$ *by a single step of constant introduction and replacement, then* R_1 *is confluent if, and only if,* R_2 *is confluent.*

If one splits the transformation into two parts (the introduction one and the replacement one), this is easy to see. Consider only introducing the rules $c \rightarrow t$ and $t \rightarrow c$. Then $\mathcal{T}(\mathcal{F})$ with the relation $\rightarrow_R^*$ is isomorphic to the congruence classes generated by $\{c \rightarrow t, t \rightarrow c\}$ on $T(\mathcal{F} \cup \{c\})$ with the relation $\rightarrow^*_{R\,\cup\,\{c \rightarrow t, t \rightarrow c\}}$. Hence this preserves confluence. Second, we have

$$\rightarrow^*_{R\,\cup\,\{s \rightarrow t, t \rightarrow s\}} \;=\; \rightarrow^*_{R'\,\cup\,\{s \rightarrow t, t \rightarrow s\}}$$

if R' is obtained from R by replacing somewhere s by t, and hence confluence is preserved as well.

By considering a polynomial sequence of such constant introduction and replacement steps, in what follows w.l.o.g. we can assume R to be built over a signature $\mathcal{F}$ with only the binary symbol f and constants, and that the sides of R have at most depth two, i.e., all sides are constants or terms of depth two $f(c_1, c_2)$ where c_1 and c_2 are constants.

3.3. Plaisted's transformation

In [14], it is shown that one can transform in polynomial time a GTRS R into another GTRS $closure(R)$ defined by

$$closure(R) = \{l \rightarrow r \mid \ l \text{ and } r \text{ are subterms of sides of } R \text{ with } l \rightarrow_R^* r\}$$

Note that this means that all subterms s of sides of R are left hand sides of rules of $closure(R)$, and also right hand sides (because $s \rightarrow_R^* s$). One can build $closure(R)$ by a simple polynomial fixpoint computation, where the fixpoint is reached after at most $|R|^2$ iterations (because there are at most $|R|$ different subterms of sides, and hence the closure contains at most $|R|^2$ rules):

$$R_0 \ = R \ \cup \{\ s \rightarrow s \mid s \text{ is a subterm of a side of } R\}$$

$$\begin{aligned} R_{i+1} \ = R_i \ &\cup \{\ s \rightarrow t \mid s \rightarrow u \in R_i \text{ and } u \rightarrow t \in R_i\} \\ &\cup \{\ f(l, r) \rightarrow f(l', r') \mid l \rightarrow l' \in R_i, \\ &\qquad r \rightarrow r' \in R_i \text{ and } f(l, r) \text{ and } f(l', r') \text{ are subterms of sides of } R\ \} \end{aligned}$$

Lemma 4 (see also [14]) *Let R be a GTRS, and let s and t be subterms of sides of R. Then $s \to t \in R_k$, for some k, if, and only if, $s \to_R^* t$.*

Proof: Clearly, if $s \to t$ is in some R_k, then $s \to_R^* t$. The reverse is proved by induction on: (i) the number of steps in $s \to_R^* t$ and (ii) on $|s|$. If $s \to_R^* t$ in zero steps then t is s and $s \to t$ is in R_0. Otherwise, consider a shortest derivation $s \to_R^* t$. If some rewrite step is done at the top (i.e., at position λ), then it is of the form $s \to_R^* l \to_R r \to_R^* t$ where $l \to r$ is in R_0. Moreover, by induction hypothesis $s \to l$ and $r \to v$ are in R_k for some k. Therefore $s \to t$ will be added to the sequence of R_i's, too. Suppose now that no rewrite step is done at the top. Then s and t are of the form $f(l, r)$ and $f(l', r')$, respectively, and $l \to_R^* l'$ and $r \to_R^* r'$ in a less or equal number of steps, and both $|l|$ and $|r|$ are smaller than $|s|$. Again by induction hypothesis we have that $l \to l'$ and $r \to r'$ are in R_k for some k, and therefore also $s \to t$ is. $\Box$

The relations $\to_R^*$ and $\to_{closure(R)}^*$ coincide. Hence $\to_R$ is confluent if, and only if, $\to_{closure(R)}$ is confluent. Since $closure(R)$ can be computed from R in polynomial time, and $closure(closure(R)) = closure(R)$, from now on w.l.o.g. we simply consider that R is already equal to $closure(R)$.

Definition 5 *Consider a derivation $s_0 \to_R^* s_n$ where each rewrite step $s_i \to_R s_{i+1}$ is done at a position p_i. The derivation is called* increasing *if $0 \leq i < j < n$ implies that p_j is not a prefix of p_i, i.e., no step takes place at, or above, a position of some previous step.*

For arbitrary R not necessarily $s \to_R^* t$ implies that $s \to_R^* t$ by a increasing derivation. For example, if R consists of the rules $f(a, a) \to b$ and $b \to a$, then $f(a, b) \to_R^* b$ but there is no increasing derivation $f(a, b) \to_R^* b$. However, in the following lemma we show that, starting from a side of R, under our assumption that $R = closure(R)$, increasing derivations always exist (again here we re-use ideas of [14]). Along the remaining sections we rely at several points on this fact.

Lemma 6 *Let s be a side of R. If $s \to_R^* t$, then there is an increasing derivation $s \to_R^* t$.*

Proof: Consider the shortest derivation $s \to_R^* t$. We proceed by induction on: (i) the number of steps in this derivation and (ii) on $|s|$. We distinguish two cases:
1. If some step is done at the top, then the derivation is of the form $s \to_R^* l \to_R r \to_R^* t$, where $l \to_R r$ is the last step at the top. But then, since $R = closure(R)$ and both s and r are sides of R, the rule $s \to r$ is in R, and hence, by minimality of the derivation, the derivation must be of the form $l \to_R r \to_R^* t$, where after r there are no more steps at the top. Therefore r and t are of the form $f(u, v)$ and $f(u', v')$, respectively and $u \to_R^* u'$ and and $v \to_R^* v'$ in one step less (or even less) than $s \to_R^* t$. Then, by the induction hypothesis, there are increasing derivations $u \to_R^* u'$ and $v \to_R^* v'$, which, appended to $l \to_R r$, altogether gives us an increasing derivation $s \to_R^* t$.
2. If no step is done at the top, then s and t are of the form $f(u, v)$ and $f(u', v')$, respectively. and $u \to_R^* u'$ and and $v \to_R^* v'$ in at most the same number of steps as in $s \to_R^* t$. Then, by the induction hypothesis, since $|u| < |s|$ and $|v| < |s|$, there are increasing derivations $u \to_R^* u'$ and $v \to_R^* v'$, which appended give us an increasing derivation $s \to_R^* t$. $\Box$

4. Reachability, Joinability and Stability analysis

Definition 7 *A term t is* reachable *from s (by R) if $s \to_R^* t$. The* reachability problem *is: Instance: terms s, t, a GTRS R; Problem: is t reachable from s?*

Lemma 8 *The reachability problem is decidable in polynomial time.*

Proof: This result (as well as the one below on joinability) is well-known (see e.g., [14], and also the automata-based approaches), but in order to make this paper more self-contained, let us mention that it is a corollary of Lemma 4: if we want to decide whether t is reachable from s, obviously, it suffices to add trivial rules $u \to u$ to R_0, for all subterms u of s and of t. $\Box$

Definition 9 *Two terms s and t are* joinable *if there exists some u such that $s \to_R^* u$ and $t \to_R^* u$. The* joinability problem *is: Instance: terms s and t and a GTRS R; Problem: are s and t joinable?*

Lemma 10 *The joinability problem is decidable in polynomial time.*

Proof: By a similar fixpoint construction and induction as Lemmas 4 and 6. $\Box$

Definition 11 *In a term $f(s, t)$, the root symbol f is* stable *(by R) if all terms reachable from $f(s, t)$ are rooted by f. The* stability problem *is: Instance: a term $f(s, t)$ and a GTRS R; Problem: is f stable in $f(s, t)$ by R?*

Lemma 12 *The stability problem is decidable in polynomial time.*

Proof: Since we can assume that f is the only non-constant symbol, in our setting this problem amounts to the question whether the given $f(s, t)$ rewrites into a constant, i.e., it amounts to n reachability problems, where n is the number of constants in R. $\Box$

Definition 13 *The root symbol f of a term $f(s,t)$ can be* stabilized *by R if $s \to_R^* s'$ and $t \to_R^* t'$ for some s' and t' such that f is stable in $f(s',t')$.*
The stabilization problem *is: Instance: a term $f(s,t)$ and a GTRS R; Problem: can f be stabilized in $f(s,t)$ by R?*

Example 14 *Let R be*

$$\{f(a,a) \to b; b \to f(a,b); f(a,a) \to a, a \to b, a \to a\}$$

f is stable in $f(a,b)$ and f is stabilizable in $f(a,a)$.

We state the following result here only for sides of R, because it is simpler and we do not need more for our purposes, but a similar construction works for the general stabilization problem.

Lemma 15 *The stabilization problem is decidable in polynomial time for sides $f(a,b)$ of R.*

Proof: Consider a fixpoint construction, where $f(a,b)$ always denotes a side of R, and (possibly primed) a and b denote constants:

$$\begin{array}{ll} ST_0 = & \{\ f(a,b) \mid a \to_R^* a',\ b \to_R^* b' \\ & \qquad \text{and } f \text{ is stable in } f(a',b')\ \} \\ \text{and} & \\ ST_{i+1} = & ST_i \\ \cup\ \{\ f(a,b) \mid & a \to_R^* f(a',b') \text{ and } f(a',b') \in ST_i\ \} \quad (i) \\ \cup\ \{\ f(a,b) \mid & b \to_R^* f(a',b') \text{ and } f(a',b') \in ST_i\ \} \quad (ii) \end{array}$$

This construction is clearly polynomial (at most the number of sides of R iterations). Furthermore, since rules in R have depth at most 2, clearly if $f(a,b) \in ST_i$ for some i then f can be stabilized in $f(a,b)$. Reversely, assume we have $f(a,b)$ such that a and b are sides of R and $a \to_R^* s$ and $b \to_R^* t$ and f is stable in $f(s,t)$. We prove by induction on $|f(s,t)|$ that then $f(a,b) \in ST_k$ for some k. If s and t rewrite into constants a' and b' respectively, then, since f is stable in $f(s,t)$ it is also stable in $f(a',b')$ and hence $f(a,b) \in ST_0$, because $a \to_R^* a'$ and $b \to_R^* b'$. If s and t do not both rewrite into constants, then at least one of them is rooted by a stable f. If s is rooted by a stable f then, since $a \to_R^* s$, the last step at position λ in $a \to_R^* s$ produces a side of R of the form $f(a',b')$. Since $f(a',b') \to_R^* s$, the f in $f(a',b')$ can be stabilized, and since $|s| < |f(s,t)|$, by induction hypothesis we have that $f(a',b')$ is in some ST_i. Therefore, since $a \to_R^* f(a',b')$, by (i) $f(a,b)$ is in ST_{i+1}. Similarly, if t is rooted by a stable f then by (ii) $f(a,b)$ is in ST_{i+1}. $\Box$

5. Some Necessary Conditions for Confluence

From now on we will consider a concrete GTRS R to which all the transformations of Section 3 have been applied, and we use the following terminology.

Definition 16 *Let s be a side of R.*

- *$Topsteps(s)$ is the set of terms reachable from s by one rewrite step at position λ, followed, if the term obtained is not a constant, by two more steps at positions 1 and 2.*

  Note that, by the closure transformation of Section 3.3, in R there are trivial rules $l \to l$ for all subterms of sides of R, and that $Topsteps(s)$ is the set of terms t that are sides of R, or f applied to sides of R, such that $s \to_R^* t$.

- *$Left(s)$ is the subset of $Topsteps(s)$ with terms $f(l,r)$ where l is rooted by f and f can be stabilized in l*

- *$Right(s)$ is the subset of $Topsteps(s)$ with terms $f(l,r)$ where r is rooted by f and f can be stabilized in r*

Example 17 *Let R be*

$$\{\ a \to f(b,b),\ b \to f(a,a),\ f(a,b) \to a\ \}$$

plus the five reflexive rules. Then by the closure transformation, the rule $f(a,b) \to f(b,b)$ is added, and we have:
$$Topsteps(a) = \left\{ \begin{array}{c} a,\ f(b,b),\ f(b,f(a,a)), \\ f(f(a,a),b),\ f(f(a,a),f(a,a)) \end{array} \right\},$$
$Left(a) = \{\ f(f(a,a),b),\ f(f(a,a),f(a,a))\ \}$, and
$Right(a) = \{\ f(b,f(a,a)),\ f(f(a,a),f(a,a))\ \}$.

Definition 18 *Let s be a side of R. The* confluence conditions *for s (in short,* C-conditions *for s) are the following:*

1. *all the terms in $Topsteps(s)$ are pairwise joinable*
2. *for all $f(l_1,r_1)$ and $f(l_2,r_2)$ in $Left(s)$, the terms r_1 and r_2 are joinable;*

   analogously happens for $Right(s)$, l_1 and l_2
3. *if $Left(s) \neq \emptyset$, for all $t \in Topsteps(s)$, there exists $t' \in Left(s)$ reachable from t;*

   analogously happens for $Right(s)$
4. *if $Left(s) \neq \emptyset$, for all $f(l,r)$ in $Right(s)$, there exists l' reachable from l such that $f(l',r)$ is in $Left(s)$*

   if $Right(s) \neq \emptyset$, for all $f(l,r)$ in $Left(s)$ exists r' reachable from r such that $f(l,r')$ is in $Right(s)$

A rewrite system R is said to satisfy the C-conditions if all its sides do.

Example 19 *(Example 17 continued) All C-conditions are satisfied for a, b, and $f(a,b)$, except that the terms $f(f(b,b),f(a,a))$ and $f(f(a,a),f(a,a))$ that are both in $Right(f(a,b))$ have non-joinable first arguments and hence, by the following lemma, R is not confluent.*

Lemma 20 *The C-conditions for s are necessary for confluence of R.*

Proof:

1. If R is confluent, all terms reachable from s are joinable.

2. If $f(l_1, r_1)$ and $f(l_2, r_2)$ are in $Left(s)$, there exist l_1' and l_2' reachable from l_1 and l_2 and rooted by a stable f. If R is confluent, then the terms $f(l_1', r_1)$ and $f(l_2', r_2)$, that are reachable from s, must be joinable. Since no more steps at position λ are possible on them because rules have depth at most two, r_1 and r_2 must be joinable.

3. If $f(l, r) \in Left(s)$, there exists an l' reachable from l where l' is rooted with a stable f. If R is confluent, each t in $Topsteps(s)$ is joinable with $f(l', r)$, and hence t rewrites into some $f(l'', r'')$ where l'' is rooted with a stable f. Since t is in $Topsteps(s)$, either t is a constant or it is f applied to two sides of R. In both cases, in a derivation $t \to_R^* f(l'', r'')$ there is always a last term $f(t_1, t_2)$ such that t_1 and t_2 are sides of R, and there is also a t_1' that is the last side of R appearing at position 1 of a term in this derivation. Then $t_1 \to_R^* t_1'$, and hence $t \to_R^* f(t_1', t_2)$, where $f(t_1', t_2)$ is in $Topsteps(s)$. Moreover, $t_1' \to_R^* l''$, and hence t_1' is rooted by an f that can be stabilized in t_1'. Therefore $f(t_1', t_2)$ is a term in $Left(s)$ that is reachable from t.

4. Consider $f(l_1, r_1)$ in $Left(s)$ and $f(l_2, r_2)$ in $Right(s)$. Then there exist l_1' and r_2' reachable from l_1 and r_2 respectively, where l_1' and r_2' are rooted by a stable f. Since no more steps at the top are possible on $f(l_1', r_1)$ and $f(l_2, r_2')$, and R is confluent, l_1' and l_2 are joinable. Therefore there is an (increasing) derivation from l_2 into a term $f(t, t')$ rooted by a stable f. The first term l_2' in this derivation after which no more steps take place at position λ is a side of R rooted by an f that can be stabilized. Therefore, there exists an l_2' reachable from l_2 such that $f(l_2', r_2)$ is in $Left(s)$. Analogously, there exists an r_1' reachable from r_1 such that $f(l_1, r_1')$ is in $Right(s)$. □

6. Polynomial Confluence Test

From now on, we assume that R satisfies the C-conditions.

Lemma 21 *Let s be a side of R rooted by an f that can be stabilized in s. Then, for all s' reachable from s, there exist an s'' reachable from s' with f stable at its top.*

Proof: If s' is rooted by a stable f, then, taking s'' as s' the result follows. Otherwise, a constant side c of R is reachable from s', and moreover, this c is in $Topsteps(s)$. Now we distinguish two cases.

- $Left(s)$ and $Right(s)$ are empty. We know s is rooted by an f that can be stabilized, i.e., s is of the form $f(l, r)$, and there exist l' and r' with $l \to_R^* l'$ and $r \to_R^* r'$, such that f is stable in $f(l', r')$. Moreover, since $Left(s)$ and $Right(s)$ are empty, l' and r' are not rooted by a stable f. Consequently, there exist constants a and b of R such that $l' \to_R^* a$ and $r' \to_R^* b$. Now, $f(a, b)$ is in $Topsteps(s)$ since a and b are sides of R. Furthermore, f is stable in $f(a, b)$ since it is stable in $f(l', r')$ and $f(l', r') \to_R^* f(a, b)$. Moreover, c and $f(a, b)$ are joinable, i.e., they both rewrite into some s'', since s satisfies C-condition 1. Altogether, we have $s' \to_R^* c \to_R^* s''$ and moreover, f is stable in s'' since it is stable in $f(a, b)$.

- One of $Left(s)$ or $Right(s)$ is non-empty, w.l.o.g., say $Left(s)$. Since s satisfies C-condition 3, there exists an $f(l, r)$ in $Left(s)$ reachable from c. Since $f(l, r) \in Left(s)$, l is headed by an f that can be stabilized, and hence there exists an l' reachable from l rooted by a stable f. Therefore, $f(l', r)$ is the s'' we wanted to obtain: we have $s' \to_R^* c \to_R^* f(l', r)$, and f is stable in $f(l', r)$. □

Definition 22 *Two terms s and t are* deeply joinable *if s' and t' are joinable whenever $s \to_R^* s'$ and $t \to_R^* t'$ (i.e., if each term reachable from s is joinable with each term reachable from t).*

Example 23 *Let $R = \{a \to c, b \to c, a \to f(e, e), b \to f(d, d), d \to f(e, e), e \to f(d, d)\}$. a and b are joinable, but not deep joinable since $f(d, d)$ and $f(e, e)$ are not joinable.*

Definition 24 *Let s_1 and s_2 be sides of R. The* deep joinability conditions *(*DJ-conditions *in short) for s_1 and s_2 are the following:*

1. *if t_1 in $Topsteps(s_1)$ and t_2 in $Topsteps(s_2)$ then t_1 and t_2 are joinable*

2. *$Left(s_1) = \emptyset$ iff $Left(s_2) = \emptyset$, and $Right(s_1) = \emptyset$ iff $Right(s_2) = \emptyset$*

3. *if $f(l_1, r_1) \in Left(s_1)$ and $f(l_2, r_2) \in Left(s_2)$ then r_1 and r_2 are joinable*

4. *if $f(l_1, r_1) \in Right(s_1)$ and $f(l_2, r_2) \in Right(s_2)$ then l_1 and l_2 are joinable*

Lemma 25 *The DJ-conditions for s and s' are necessary conditions for deep joinability of s and s'.*

Lemma 25 is proved by arguments similar to the ones used in lemma 20.

Lemma 26 *Let s_1 and s_2 be sides of R.*

If $f(l_1, r_1) \in Left(s_1)$ and $f(l_2, r_2) \in Left(s_2)$ and l_1 and l_2 are not deeply joinable, then s and s' are not deeply joinable either.

If $f(l_1, r_1) \in Right(s_1)$ and $f(l_2, r_2) \in Right(s_2)$ and r_1 and r_2 are not deeply joinable, then s and s' are not deeply joinable either.

Proof: We only prove the first statement; the other one is analogous. Assume $f(l_1, r_1) \in Left(s_1)$ and $f(l_2, r_2) \in Left(s_2)$ and l_1 and l_2 are not deeply joinable, i.e., there exist non-joinable terms l'_1 and l'_2 such that $l_1 \to^*_R l'_1$ and $l_2 \to^*_R l'_2$. By lemma 21 there exist l''_1 and l''_2 with $l'_1 \to^*_R l''_1$ and $l'_2 \to^*_R l''_2$ such that f is stable in l''_1 and in l''_2; and l''_1 and l''_2 are not joinable since l'_1 and l'_2 are not joinable either. Therefore $s_1 \to^*_R f(l_1, r_1) \to^*_R f(l''_1, r_1)$ and $s_2 \to^*_R f(l_2, r_2) \to^*_R f(l''_2, r_2)$, and $f(l''_1, r_1)$ and $f(l''_2, r_2)$ are not joinable, since f is stable in l''_1 and in l''_2, and no rewrite at the root is possible. Hence s_1 and s_2 are not deeply joinable. □

Lemma 27 *Let s_1 and s_2 be sides of R that satisfy the DJ-conditions, and such that $Left(s_1)$, $Left(s_2)$, $Right(s_1)$ and $Right(s_2)$ are empty. Then s_1 and s_2 are deeply joinable.*

Proof: We have to prove that for all s'_1 and s'_2 reachable from s_1 and s_2 respectively, we have that s'_1 and s'_2 are joinable. Since s_1 and s_2 satisfy DJ-condition 1, it suffices to show that some term in $Topsteps(s_1)$ is reachable from s'_1, and analogously for s'_2. If s'_1 is a constant, then it is in $Topsteps(s_1)$. Otherwise s'_1 is of the form $f(l, r)$. Since $Left(s_1)$ and $Right(s_1)$ are empty, neither l nor r is rooted by a stable f. Therefore $l \to^*_R a$ and $r \to^*_R b$ for constants a and b. Hence $s'_1 = f(l, r) \to^*_R f(a, b)$, that is in $Topsteps(s_1)$ since a and b are sides of R. □

Theorem 28 *The confluence problem for a GTRS is decidable in polynomial time.*

Proof: It is well-known that a GTRS R is confluent if, and only if, all its left hand sides l are deeply joinable with themselves, i.e., if R is confluent from each left hand side. Consider the following fixpoint construction, with the intuitive meaning that a pair (s, s') of sides of R is included in some NDJ_k whenever s and s' are not deeply joinable. In the following, s and s' always denote sides of R:

$NDJ_0 = \{\ (s, s') \mid s, s'$ do not satisfy the DJ-conditions$\}$

and

$$
\begin{aligned}
NDJ_{i+1} \;=\; & NDJ_i \\
& \cup \{\ (s, s') \mid f(l, r) \in Left(s), \\
& \qquad f(l', r') \in Left(s'),\quad (l, l') \in NDJ_i\ \} \\
& \cup \{\ (s, s') \mid f(l, r) \in Right(s), \\
& \qquad f(l', r') \in Right(s'),\ (r, r') \in NDJ_i\ \}
\end{aligned}
$$

This construction is clearly polynomial (at most n^2 such pairs (s, s') exist where n is the number of subterms of sides of R). Furthermore, whenever $(s, s') \in NDJ_k$ for some k, we have that s and s' are not deeply joinable, by induction on i using Lemma 26 (with Lemma 25 for the base case).

We now show the reverse implication: we prove that, for all pairs (s_1, s_2) of sides of R, for all t_1, t_2, if $s_1 \to^*_R t_1, s_2 \to^*_R t_2$ and t_1, t_2 are not joinable, then (s_1, s_2) belongs to some NDJ_k. We proceed by induction on $max(height(t_1), height(t_2))$, assuming without loss of generality (by lemma 6) that the derivation from s_1 to t_1 and from s_2 to t_2 are increasing.

If $max(height(t_1), height(t_2)) < 3$, then $s_i \to^*_R t_i$ implies that t_i is reachable from s_i in at most three steps, by increasingness of the derivations. It follows that $t_1 \in Topsteps(s_1)$ and $t_2 \in Topsteps(s_2)$ and hence s_1 and s_2 do not satisfy the DJ-conditions (otherwise by DJ-condition 1, t_1 and t_2 would be joinable) and therefore $(s_1, s_2) \in NDJ_0$.

For the remaining cases, we assume $max(height(t_1), height(t_2)) \geq 3$ and that s_1 and s_2 satisfy the DJ-conditions. By DJ-condition 2, $Left(s_1) = \emptyset$ iff $Left(s_2) = \emptyset$, and $Right(s_1) = \emptyset$ iff $Right(s_2) = \emptyset$. Moreover, not all of them are empty (otherwise by Lemma 27 s_1 and s_2 would be deeply joinable). Assume $Left(s_1) \neq \emptyset$ and $Left(s_2) \neq \emptyset$. (We omit the symmetric case where the $Right$'s are non-empty.)

Below we prove the following statement, which holds for $i = 1, 2$.

There is an $f(l_i, r_i) \in Left(s_i)$ and an $f(l'_i, r'_i)$ with $t_i \to^*_R f(l'_i, r'_i)$ such that:

1. $l_i \to^*_R l'_i$ and $r_i \to^*_R r'_i$
2. $height(f(l'_i, r'_i)) \leq 3$ or $height(f(l'_i, r'_i)) = height(t_i)$
3. If $Right(s_i) = \emptyset$ then $r'_i = r_i$ and otherwise $f(l_i, r_i) \in Right(s_i)$

Assume we apply this statement to s_i and t_i, for $i = 1, 2$, obtaining $f(l_i, r_i)$ and $f(l'_i, r'_i)$. Then we complete the proof as follows. Since $t_1 \to^*_R f(l'_1, r'_1)$ and $t_2 \to^*_R f(l'_2, r'_2)$, and t_1 and t_2 are not joinable, we know that l'_1 and l'_2 are not joinable or r'_1 and r'_2 are not joinable. Moreover, by the above statement 2,

$$max(height(l'_1), height(l'_2)) < max(height(t_1), height(t_2))$$

and

$$max(height(r'_1), height(r'_2)) < max(height(t_1), height(t_2)).$$

Since, by increasingness, l_1, l_2, r_1, r_2 are sides of R, we can apply the induction hypothesis to either (l_1, l_2) or (r_1, r_2) (choose the non-joinable pair), i.e. either $(l_1, l_2) \in NDJ_k$ or $(r_1, r_2) \in NDJ_k$. If $(l_1, l_2) \in NDJ_k$ then $(s_1, s_2) \in NDJ_{k+1}$. If $(r_1, r_2) \in NDJ_k$ and $Right(s_1) = Right(s_2) = \emptyset$ then $(s_1, s_2) \in NDJ_{k+1}$. If $(r_1, r_2) \in NDJ_k$ and $Right(s_1) = Right(s_2) \neq \emptyset$ then r'_1 is r_1 and r'_2 is r_2 and hence r_1 and r_2 are not joinable; then by DJ-condition 3, and since $f(l_1, r_1) \in Left(s_1)$ and $f(l_2, r_2) \in Left(s_2)$, again we have $(s_1, s_2) \in NDJ_{k+1}$. We now prove the statement, considering the cases:

t_i **is a constant.** Then $t_i \in Topsteps(s_i)$ and, by C-condition 3, $t_i \rightarrow^*_R f(l_i, r_i) \in Left(s_i)$.

If $Right(s_i) = \emptyset$, taking $l'_i = l_i$ and $r'_i = r_i$ the statement holds because $height(f(l_i, r_i)) \leq 3$ since l_i and r_i are sides of R.

If $Right(s_i) \neq \emptyset$, by C-condition 4, there is a r'_i such that $r_i \rightarrow^*_R r'_i$, $f(l_i, r'_i)$ is in $Right(s_i)$ and $Left(s_i)$. Choosing $l_i = l'_i$ again the statement holds because $height(f(l_i, r'_i)) \leq 3$ since l_i and r'_i are sides of R.

t_i **is** $f(u, v)$ **where none of** u **and** v **is headed by a stable** f**.** Then u and v rewrite into constants a and b and $t_i \rightarrow^*_R f(a, b)$ Let the first rewrite step from s_i to t_i be $s_i \rightarrow f(\alpha, \beta)$, which may be assumed to take place at position λ (since there is a rule $s_i \rightarrow s_i$ in R). Then, by increasingness of the derivation, $\alpha \rightarrow^*_R u$ and $\beta \rightarrow^*_R v$ and, since s is a side of R, α, β are constants which are sides of R. Then, by reflexive-transitive closure of R, $\alpha \rightarrow a \in R$ and $\beta \rightarrow b \in R$, which implies that $f(a, b) \in Topsteps(s_i)$. We conclude from $t \rightarrow^*_R f(a, b) \rightarrow^*_R f(l, r) \in Left(s)$ as in the previous case.

t_i **is** $f(u, v)$**,** u **is headed by a stable** f **and** v **is not.** Then v rewrites into a constant a and, as before, there is an increasing derivation $s \rightarrow^*_R f(u, a)$, where w.l.o.g. the first step is at position λ. Assume this first step gives $f(c, d)$. Then $d \rightarrow_R a$ in one step (since d and a are sides), obtaining $f(c, a)$. Since $c \rightarrow^*_R u$, by one more step on c, we obtain as the fourth term in this derivation $f(l_i, a)$, where l_i is headed by an f that can be stabilized, since $l_i \rightarrow^*_R u$. Hence $f(l_i, a) \in Left(s_i)$. If $Right(s_i) = \emptyset$, choosing $l'_i = u$ and $r'_i = r_i = a$ the statement holds. If $Right(s_i) \neq \emptyset$, by C-condition 4, $a \rightarrow^*_R r'_i$ such that $f(l_i, r'_i)$ is in $Right(s_i)$ and $Left(s_i)$; then, choosing $l'_i = u$ and $r'_i = r_i$ the statement holds.

t_i **is** $f(u, v)$**,** v **is headed by a stable** f **and** u **is not.** Then u rewrites into a constant a and there is an increasing derivation $s_i \rightarrow^*_R f(a, v)$ where, analogously to the previous case, the fourth term is of the form $f(a, r_i)$, where r_i is headed by an f that can be stabilized, since $r_i \rightarrow^*_R v$. Hence $f(a, r_i) \in Right(s_i)$. Then by C-condition 4, there exists some l_i reachable from a such that $f(l_i, r_i)$ is in $Left(s_i)$ and in $Right(s_i)$. Then we have $f(u, v) \rightarrow^*_R f(a, v) \rightarrow^*_R f(l_i, v)$. Since $Right(s_i) \neq \emptyset$, by choosing $l'_i = l_i$ and $r'_i = v$, the result follows.

t_i **is** $f(u, v)$ **and both** u **and** v **are headed by a stable** f**.** Then there is an increasing derivation $s_i \rightarrow^*_R t_i$ where, as in the previous two cases, we can assume that the fourth term is $f(l_i, r_i)$ and where both l_i and r_i are headed by an f that can be stabilized, since $l_i \rightarrow^*_R u$ and $r_i \rightarrow^*_R v$. Since $Right(s_i) \neq \emptyset$, choosing $l'_i = u$ and $r'_i = v$ the statement holds. □

7. PTIME-hardness

The following result completely determines the complexity classes of confluence problems for the different kinds of GTRS.

Theorem 29 *The following problem is PTIME-hard.*
Instance: a GTRS R built over only unary symbols and constants
Question: is R confluent?

Proof: We reduce the alternating graph reachability problem [2]:

Given: A directed graph $G = (V, E)$, a partition $V = A \cup B$ of the vertices and two designated vertices s, t.

Problem: Is t reachable from s in the alternating graph ?

t is reachable from s in the alternating graph if either $s = t$ or $s \in A$ (universal vertices) and for every edge $(s, u) \in E$, t is reachable from u in the alternating graph, or $s \in B$ (existential vertices) and there is an edge $(s, u) \in E$ such that t us reachable from u in the alternating graph.

First, we may assume that the outdegree of every vertex is at most 2 (by a LOGSPACE reduction) and we may also assume that there is no edge departing from the target vertex t.

Then we associate one unary symbol with each vertex. Let t be the symbol associated with the target and s be the symbol associated with the source.

For every vertex $a \in B$, $a \neq t$ with successors b, c, we build the rules

$$\begin{array}{rclcrcl} a(0) & \rightarrow & b(c(0)) & \quad & b(c(0)) & \rightarrow & a(0) \\ a(0) & \rightarrow & c(b(0)) & & c(b(0)) & \rightarrow & a(0) \end{array}$$

For every vertex $a \in A$, $a \neq t$ with successors b, c, we build the rules

$$\begin{array}{rclcrcl} a(0) & \rightarrow & b(A(c(0))) & \quad & b(A(c(0))) & \rightarrow & a(0) \\ a(0) & \rightarrow & c(A(b(0))) & & c(A(b(0))) & \rightarrow & a(0) \end{array}$$

Where A is an additional unary symbol.

For every vertex a with only one successor b, add the symmetric rules $a(0) \rightarrow b(0)$. Finally, for every $a \neq A$, we add the rules

$$\begin{array}{rclcrcl} a(t(0)) & \rightarrow & t(0) & \quad & t(0) & \rightarrow & a(t(0)) \\ a(A(t(0))) & \rightarrow & a(0) & & a(0) & \rightarrow & a(A(t(0))) \end{array}$$

The rewrite system can be constructed in LOGSPACE and it is confluent since all rules are bi-directional. Moreover, t is reachable from a vertex a in the alternating graph if and only if $a(0) \rightarrow^* t(0)$. (We skip the details of this proof here).

Now, add the rules $s'(0) \rightarrow t(0)$ and $s'(0) \rightarrow s(0)$. The confluence is equivalent to $s(0) \rightarrow^* t(0)$ which, in turn, is equivalent to the reachability of t from s in the alternating graph. $\square$

8. Conclusions

The algorithmic questions for subclasses of rewrite systems are either undecidable or untractable. The hope was that ground rewrite system would be an exception. Indeed, after reachability, joinability [3, 13], termination [14], word problem [10], remained confluence. We have shown here that this last problem is also decidable in polynomial time.

Our proof of this result is based on the fact that in polynomial time one can decide deep joinability, i.e., "given s, t, R, are all terms reachable from s joinable with all the ones reachable from t?". This is very surprising. For instance, the very similar "half deep joinability" question: "given s, t, R, is any term reachable from s joinable with t?" is EXPTIME-complete, although at first sight it looks simpler since instead of checking all descendants of t one only needs to consider t itself.

Lemma 30 *The half deep joinability problem is EXPTIME-hard.*

Proof: By reducing the well-known EXPTIME-complete problem whether a given tree automaton A accepts all ground terms over the given signature $\mathcal{F}$, i.e., all terms in $\mathcal{T}(\mathcal{F})$ [16]. Let R_A be the GTRS consisting of the transition rewrite rules of A written in a "generating" way, like $q \rightarrow f(q_1, \ldots, q_n)$, and the rules

$$\begin{array}{lll} 1 & \rightarrow G & \\ G & \rightarrow f(G, \ldots, G) & \text{(one rule for each } n\text{-ary } f \text{ in } \mathcal{F}) \\ G & \rightarrow c & \text{(one rule for each constant } c \text{ in } \mathcal{F}) \\ 1 & \rightarrow F & \\ F & \rightarrow q_f & \text{(one rule for each accepting state } q_f \text{ of } A) \end{array}$$

where 1, F, and G are new symbols not in $\mathcal{F}$.

With R_A every term reachable from 1 is joinable with F if, and only if, A accepts $\mathcal{T}(\mathcal{F})$. First, note the following facts: $R(1) = R(F) \cup R(G)$, $R(G) = T(\mathcal{F} \cup \{G\})$, $R(G) \supset \mathcal{T}(\mathcal{F})$, $R(F) \cap \mathcal{T}(\mathcal{F}) = L(A)$, and all terms in $\mathcal{T}(\mathcal{F})$ are in normal form. Now, if every term reachable from 1 is joinable with F then every term reachable from G is joinable with F, and hence every term in $\mathcal{T}(\mathcal{F})$ is joinable with F. Since every term in $\mathcal{T}(\mathcal{F})$ is in normal form, it must be reachable from F and hence it must be in $L(A)$, i.e., $L(A) = \mathcal{T}(\mathcal{F})$. Reversely, suppose $L(A) = \mathcal{T}(\mathcal{F})$. Then, since every term reachable from G rewrites into a term in $\mathcal{T}(\mathcal{F})$, every term reachable from G is joinable with F, and hence every term reachable from 1 is joinable with F. $\square$

Our polynomial-time decision procedure for confluence is also surprising because confluence amounts to half deep joinability of l and t, for all left hand sides l and all t reachable in one step from l, i.e., these tests can be individually very hard, although their conjunction is polynomial. In general, for individual "symmetric" questions like deep joinability, more efficient techniques appear to exist. For example, we believe that a deep joinability tree automaton for a GTRS R can be built in polynomial time.

Our result suggests that there is a non-trivial class of (tree) automata for which inclusion (and possibly other decision problems) can be decided in polynomial time, while it is EXPTIME-hard in general. This could be extremely useful in deriving efficient decision techniques for logical problems which are expressible in this class. This idea is supported by the following remarks: 1) (tree) automata are a particular case of ground rewrite systems; 2) while the inclusion problem $R^{-1} \circ R \subseteq R \circ R^{-1}$, i.e. the confluence of R, is EXPTIME-complete for both recognizable binary relations R and products of (unary) recognizable relations, we have shown that it is polynomial for relations defined by ground tree transducers. (For the $R^{-1} \circ R \subseteq R \circ R^{-1}$, the EXPTIME membership is obtained by complementing the right side and checking for emptiness the intersection with the lft side. The EXPTIME-hardness is obtained by reduction of the universality problem for tree automata [16]: for instance consider for R the relation $L(A) \times \mathcal{T}(\mathcal{F})$. The inclusion holds iff $L(A) = \mathcal{T}(\mathcal{F})$.)

For simplicity, we have not included a more precise complexity analysis of the different steps of our algorithm. Let us only mention here that the two first transformations in the worst case duplicate the size of R; then in a dynamic programming approach, one can build a table where for each pair of sides one stores reachability (i.e., the third transformation) and (deep) joinability and another table with, for each side s, the sets $Topsteps(s)$, $Left(s)$, and $Right(s)$. According to a more detailed analysis, this leads to an overall $|R|^5$-algorithm.

Concerning other directions for future work, we currently believe that the transformation of Section 3.2 can be eliminated, and then our techniques can be adapted for some classes of non-ground term rewrite systems. In particular, this could lead to a co-NP algorithm for the confluence of left-linear right-ground TRS (i.e., in each rule $l \to r$, no variable occurs more than once in l, and r is ground) and decidability for the confluence of right-ground TRS (two other well-known open problems).

References

[1] R. Book and F. Otto. *String Rewriting Systems*. Springer-Verlag Inc., New York, NY, USA, 1993.

[2] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, Jan. 1981.

[3] M. Dauchet, T. Heuillard, P. Lescanne, and S. Tison. Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems. *Information and Computation*, 88(2):187–201, Oct. 1990.

[4] M. Dauchet, S. Tison, T. Heuillard, and P. Lescanne. Decidability of the confluence of ground term rewriting systems. In *Proceedings, Symposium on Logic in Computer Science*, pages 353–359, Ithaca, New York, 22–25 June 1987. The Computer Society of the IEEE.

[5] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 6, pages 244–320. Elsevier Science Publishers B.V., Amsterdam, New York, Oxford, Tokyo, 1990.

[6] N. Dershowitz and D. Plaisted. Rewriting. In J. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers and MIT Press, 2001.

[7] P. J. Downey, R. Sethi, and R. E. Tarjan. Variations on the common subexpressions problem. *J. of the Association for Computing Machinery*, 27(4):758–771, 1980.

[8] D. Kapur, P. Narendran, and F. Otto. On ground confluence of term rewriting systems. *Inform. Comput.*, 86(1):14–31, 1990.

[9] D. Knuth and P. Bendix. Simple word problems in universal algebras. In *J. Leech, ed., Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.

[10] D. Kozen. Complexity of finitely presented algebras. In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing*, pages 164–177, Boulder, Colorado, 2–4 May 1977.

[11] G. Nelson and D. C. Oppen. Fast decision procedures bases on congruence closure. *Journal of the Association for Computing Machinery*, 27(2):356–364, Apr. 1980.

[12] F. Otto. On the connections between rewriting and formal language theory. In P. Narendran and M. Rusinowitch, editors, *Tenth International Conference on Rewriting Techniques and Applications (RTA)*, LNCS 1631, pages 332–355, Trento, Italy, July 2–4, 1999. Springer-Verlag.

[13] M. Oyamaguchi. The Church-Rosser property for ground term-rewriting systems is decidable. *Theoretical Computer Science*, 49(1):43–79, 1987.

[14] D. Plaisted. Polynomial time termination and constraint satisfaction tests. In C. Kirchner, editor, *5th International Conference on Rewriting Techniques and Applications (RTA)*, LNCS 690, pages 405–420, Montreal, Canada, 1993.

[15] RTA-LOOP. Problem #12, posed by Wayne Snyder in 1991. Int. Conf. on Rewriting Techniques and Applications, The list of open problems, 2001. Maintained at `http://www.lri.fr/~rtaloop/` (by R. Treinen).

[16] H. Seidl. Deciding equivalence of finite tree automata. *SIAM Journal on Computing*, 19(3):424–437, June 1990.

[17] R. E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31(1):1–12, Jan. 1984.