Algorithms and Reductions for Rewriting Problems

Rakesh M. Verma, Michaël Rusinowitch, Denis Lugiez

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Algorithms and Reductions for Rewriting Problems

Rakesh M. Verma, Michael Rusinowitch, Denis Lugiez

N° 3258

Septembre 1997

THÈME 2

ISSN 0249-6399

Algorithms and Reductions for Rewriting Problems

Rakesh M. Verma, Michael Rusinowitch, Denis Lugiez

Thème 2 — Génie logiciel
et calcul symbolique
Projet protheo

Rapport de recherche n° 3258 — Septembre 1997 — 21 pages

Abstract: In this paper we initiate a systematic study of polynomial-time reductions for some basic decision problems of rewrite systems. We then give a polynomial-time algorithm for Unique-normal-form property of ground systems for the first time. Next we prove undecidability of these problems for string rewriting using our reductions. Finally, we prove partial decidability results for Confluence of commutative semi-thue systems. The Confluence and Unique-normal-form property are also shown Expspace-hard for commutative semi-thue systems.

Key-words: Rewriting, Complexity, Reduction

(Résumé : tsvp)

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
Téléphone : (33) 83 59 30 30 – Télécopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
Téléphone : (33) 87 20 35 00 – Télécopie : (33) 87 76 39 77

Algorithmes et Réductions en Réécriture

Résumé : Dans cet article, nous débutons une étude systématique des réductions polynomiales pour certains problémes de décision des systèmes de réécriture. Nous donnons pour la première fois un algorithme polynomial pour décider l'unicité de la forme normale pour les systèmes clos. Ensuite nous montrons l'indécidabilité des problèmes considérés pour la réécriture de mots en utilisant nos réductions. Finalement nous prouvons la décidabilité de la confluence de la réécriture pour les systèmes semi-thuens commutatifs. La confluence et la propriété d'unicité de la forme normale sont prouvées Expspace-difficile pour ces mêmes systèmes.

Mots-clé : Réécriture, Complexité, Réduction

1 Introduction

In this paper, we initiate a systematic study of reducibilities among some basic decision problems for rewrite systems. We also develop efficient algorithms and prove decidability/undecidability for some restricted versions of these problems. The importance of these problems is well-established and it is well-known that they are all undecidable in general. Surprisingly however, although these problems are well-known and well-studied, so far – to the best of our knowledge – there is no systematic attempt to establish tight relationships between these decision problems. Specifically, we are not aware of any attempts to relate them via the well-known complexity-theoretic concept of a polynomial-time reduction or transformation. The benefit of such reductions between problems is well-known and cannot be overemphasized. Using reducibilities it is possible to economically derive and express many results for these problems, rather than working from scratch each time one is confronted with one of these problems.

The problems considered here are: Reachability, Joinability, Confluence, Unique normalization, and Unique-normal-form property. We establish many polynomial-time reductions among these problems for ordinary rewriting as well as for rewriting modulo equational theories. Using these reductions and the undecidability of a restricted version of Reachability we are able to show that all these problems are undecidable for ground rewriting modulo associativity. We show that there is a polynomial-time algorithm for the Unique-normal-form property of ground rewrite systems, and for ground rewriting modulo commutativity. This result is interesting since this property may be regarded as being "close" to Confluence and there is no good upper bound for Confluence of ground systems. Using this result and polynomial reductions we are able to show membership of some of these problems in P for ground rewriting and for ground rewriting modulo commutativity. Finally, we show the decidability of confluence for rewrite rules involving one AC operator and constants. Confluence testing for ground rewrite systems modulo a single AC operator is shown to be Expspace-hard via a simple reduction from the word problem for commutative semigroups, which is proved Expspace-complete in [10].

We now discuss earlier related work on the problems of this paper. Decidability of Confluence and Reachability for ground systems was shown by Oyamaguchi [12]. Decidability of Reachability, Joinability and Confluence for left-linear, right-ground systems was shown by Dauchet et al. [3]. Decidability of Reachability and Joinability for right-ground rewrite systems was shown by Oyamaguchi [13]. Deruyver and Gilleron [5] studied the decidability of reachability for ground systems and ground rewriting modulo associativity, commutativity.

2 Preliminaries

We assume familiarity with basic notions of rewriting (see [4, 8] for excellent surveys). Let V be a countable set of elements called *variables* and Σ be a countable set of function symbols with $\Sigma \cap V = \emptyset$. $\mathcal{T}$ is the set of all terms of a first-order language constructed from V

and Σ. It is convenient to think of terms as ordered rooted trees. If $s \in \mathcal{T}$, then $Var(s)$ denotes the *set* of variables in s. A *rule* is a pair of terms $l \to r$, such that $l \notin V$ and $Var(r) \subseteq Var(l)$ (the variables in a rule are implicitly universally quantified). A *system* R is a finite set of rules. A rule $l \to r$ is *collapsing* if $r \in V$ and *noncollapsing* otherwise, it is *variable-preserving* if $Var(l) = Var(r)$. Similarly, an equation $s = t$ is collapsing if either s or t is a variable and variable-preserving if $Var(s) = Var(t)$. We extend the concepts noncollapsing and variable-preserving to sets of rules and equations in the obvious way. The notion of a *path* or *occurrence* is used to refer to subterms in a term as follows. A path is either the empty string λ that reaches the root or $o.i$ (o is a path and i an integer) which reaches the ith argument of the root of the subterm reached by o. $o \leq q$ whenever $\exists p\ o.p = q$; if $p \neq \lambda$ also, then $o < q$. t/o refers to the subterm of t reached by o. $O(t)$ denotes the set of occurrences of t.

We use $\to$ to denote rewrite relations, where $s \to_R t$ if there is a rule $l \to r$, an occurrence $o \in O(s)$, and a substitution σ such that the subterm $s/o = \sigma(l)$ and $t = s[o \leftarrow \sigma(r)]$. Let $\to$ be a relation over $\mathcal{T}$. We say that $\to$ is: *stable* iff $\forall \sigma,\ \forall s, t, s \to t \Rightarrow \sigma(s) \to \sigma(t)$; *compatible* iff $\forall A,\ \forall u \in O(A),\ \forall B, C, B \to C \Rightarrow A[u \leftarrow B] \to A[u \leftarrow C]$. Clearly the ordinary rewrite relation is the smallest compatible stable relation containing R.

Notation. We use $=_R$ to represent the least equivalence relation containing $\to_R$. We also say that $a =_R b$ is an equational proof. When the set of rules R is clear from the context, we drop the subscripts from $\to$. The reflexive-transitive closure of $\to$ is denoted by $\stackrel{*}{\to}$ (i.e., a sequence of zero or more reductions). The transitive closure of $\to$ is denoted by $\stackrel{+}{\to}$ (i.e., a sequence of one or more reductions). The symmetric closure of $\to$ is denoted by $\leftrightarrow$. We use the term *root reduction* to indicate reduction of the entire term, and *nonroot reduction* to indicate reduction at a proper subterm. A polynomial-time reduction between two decision problems is denoted $\leq_P$.

The following facts are used implicitly in the rest of the paper.

Proposition 1 (Basic facts on equational proofs) *(i) $s =_R t$ implies $\sigma(s) =_R \sigma(t)$ for any substitution σ (stability under substitution).*

(ii) $s =_R t$ implies $C[s] =_R C[t]$ for any context C (stability under contexts).

(iii) Let $s =_R t$ and let O be the set of occurrences at which reductions are carried out in this proof. If $o \in O$ is any minimal occurrence, then $s/o =_R t/o$ (projection property).

We say that relation $\to$ is *locally confluent* if and only if $\forall A, B, C\ A \to B$ and $A \to C$ implies $\exists\ D$ such that $B \stackrel{*}{\to} D$ and $C \stackrel{*}{\to} D$. We say that relation $\to$ is *confluent* (CR) if and only if $\forall A, B, C\ A \stackrel{*}{\to} B$ and $A \stackrel{*}{\to} C$ implies $\exists\ D$ such that $B \stackrel{*}{\to} D$ and $C \stackrel{*}{\to} D$.

Definition 2 *Let R be a rewrite system. A term t is a* normal form *if there is no term u such that $t \to u$. A term t* has a normal form *if there is a normal form u such that $t \stackrel{*}{\to} u$. R (or $\to_R$) is uniquely normalizing (is $UN^{\to}$) if for all terms A, B, C such that $A \stackrel{*}{\to} B$ and $A \stackrel{*}{\to} C$ and B, C are normal forms we have $B = C$. R (or $\to_R$) has unique normal forms (is $UN^{=}$) if for all normal forms A, B with $A =_R B$ we have $A = B$.*

The relation between the various properties CR, UN, and $UN^{\to}$ is given in the next lemma.

Lemma 3 *The following implications hold for every system R: $CR \Rightarrow UN^{=} \Rightarrow UN^{\rightarrow}$. The reverse implications generally do not hold.*

Proof: The proofs of both statements are standard (see, for example [8]). We include examples to show that the reverse implications do not hold. $UN^{\rightarrow} \not\Rightarrow UN^{=}$. Let $R = \{a \rightarrow b, a \rightarrow c, c \rightarrow c, d \rightarrow c, d \rightarrow e\}$. R is $UN^{\rightarrow}$, but not $UN^{=}$ since $b =_R e$ and b, e are distinct normal forms. $UN^{=} \not\Rightarrow CR$. Let $R = \{a \rightarrow b, b \rightarrow b, a \rightarrow c, c \rightarrow c\}$. R is $UN^{=}$ (a, b and c are not normal forms) but R is not CR since b and c do not have a common reduct. □

The following six decision problems are studied in this paper.
Reachability. **Instance:** Rewrite System, R, terms s, t. **Question:** Does $s \stackrel{*}{\rightarrow}_R t$?
Normal form reachability. **Instance:** Rewrite System, R, terms s, t, where t is an R normal form. **Question:** Does $s \stackrel{*}{\rightarrow}_R t$?
Joinability. **Instance:** Rewrite System, R, terms s, t. **Question:** Is there a term u such that $s \stackrel{*}{\rightarrow}_R u$ and $t \stackrel{*}{\rightarrow}_R u$?

For the three problems above, we also define versions in which the input rewrite system satisfies some property P. To indicate these versions, we attach the phrase "for P systems" to the basic problem.
Confluence/UN$^{=}$/UN$^{\rightarrow}$. **Instance:** Rewrite System R. **Question:** Is R confluent/$UN^{=}$/$UN^{\rightarrow}$?

3 Reductions

In the following whenever R is ground, then, without loss of generality, we may assume that all terms appearing in the problem instances are also ground. Note that one of the desired properties of the following reductions is the preservation of groundness of the instances.

Theorem 4 *Normal form reachability for confluent systems $\leq_P$ Joinability for confluent systems.*

Proof: Given R, s, t, where t is a normal form, we construct an instance of Joinability R', s', t' by setting $R' = R$, $s' = s$ and $t' = t$. The rest is easy. □

The following lemma will prove useful below. It means that an equational proof step remains valid when substituting a free constant by any term. We denote $A[e_1 \parallel f_1, \ldots, e_n \parallel f_n]$ the parallel replacement of all occurrences of ground terms e_i by f_i, for $i = 1, \ldots, n$.

Lemma 5 *Let E be a set of equations, let $c_1, \ldots, c_n$ be a set of constants not occurring in E, and let $A, B, s_1, \ldots, s_n$ be terms. If $A =_E B$ then $A[c_1 \parallel s_1, \ldots, c_n \parallel s_n] =_E B[c_1 \parallel s_1, \ldots, c_n \parallel s_n]$*

Theorem 6 *Joinability for confluent systems $\leq_P$ Not $UN^{=}$.*

Proof: Given R, s, t with R confluent, we construct below an instance R' of $UN^{=}$.
Case 1: R is ground. Let $R' = R \cup \{s \rightarrow c, t \rightarrow d\} \cup T = \{a \rightarrow a \mid$ for all constants $a \in R, s, t\}$, where c, d are new constants not in R, s, t. T is a set of rules that reduces every

ground term over the signature of R, s, t to itself. We must show that $n =_{R'} N$ for distinct normal forms $n \neq N$ iff $s \downarrow_R t$. The if direction is obvious. For the only if direction, observe that since R is confluent, $R \cup T$ is also confluent and hence also $UN^=$. Also because of the T rules we must have $\{n, N\} = \{c, d\}$. Therefore, by selecting a shortest length equational proof $c =_{R'} d$, we must have $s =_{R \cup T} t$ since R' is ground. Since $R \cup T$ is also confluent, we have $s \downarrow_{R \cup T} t$, which implies $s \downarrow_R t$.
Case 2: R is not ground. Without loss of generality we may assume that both s and t cannot be variables, since in this case they are never joinable. If one of s or t is a variable the corresponding rule is dropped from R' below. We may also assume that $(Var(s) \cup Var(t)) \cap Var(R) = \emptyset$. The variables of s and t become new constants of R'.

Let $R' = R \cup \{s \to c, t \to d\} \cup T'$, where $T' = T \cup \{x \to x \mid x \in Var(s) \cup Var(t)\} \cup \{h(z_1, \ldots, z_p) \to h(z_1, \ldots, z_p) \mid$ for every function symbol h of arity $p > 0$ in $R, s, t\}$. We must show that $n =_{R'} N$ for distinct normal forms n, N iff $s \downarrow_R t$. The if direction is obvious. For the only if direction observe that since R is confluent, $R \cup T'$ is also confluent and hence $UN^=$. Therefore, because of the T' rules we conclude that $\{n, N\} = \{c, d\}$. The rest is justified in the Appendix.

Finally, note that for both cases the reduction takes only logspace and polynomial time. □

Theorem 7 *Joinability for confluent systems $\leq_P$ Confluence.*

Proof: Given R, s, t with R confluent, we build the system $R' = R \cup T$ where $T = \{a \to s', a \to t'\}$, a is a new constant and s', t' are obtained by replacing every variable x of s, t by a new constant x'. We show that $s \downarrow_R t$ iff R' is confluent. For the if direction, since R' is confluent we have $s' \to_{R'} u$ and $t' \to_{R'} u$ for some u. Since no rule in R' can introduce a subterm a there are no occurrence of a in the derivation. Hence we have $s' \to_R u$ and $t' \to_R u$ and therefore $s \downarrow_R t$. For the only if direction, we shall prove that $\stackrel{*}{\to}_T \cdot \stackrel{*}{\to}_R$ is strongly confluent in Lemma 10 below. Since strong confluence implies confluence [8] we will get that $\stackrel{*}{\to}_T \cdot \stackrel{*}{\to}_R$ is confluent and since $\stackrel{*}{\to}_{R'} = (\stackrel{*}{\to}_T \cdot \stackrel{*}{\to}_R)^*$, this will finish the proof. □

Lemma 8 *If $u \stackrel{*}{\to}_T v$ and $u \stackrel{*}{\to}_T w$ then there exists v', w', h such that $v \stackrel{*}{\to}_T v' \stackrel{*}{\to}_R h$, $w \stackrel{*}{\to}_T w' \stackrel{*}{\to}_R h$, and there is no occurrence of a in v', w', h.*

Proof: The lemma is illustrated by Figure 1 (a). Let U be the set of positions of a in u. Take $v' = v[a \parallel s']$, $w' = w[a \parallel s']$. Note that v' and w' may differ only below occurrences $\epsilon \in U$, and for all $\epsilon \in U$, $v'/\epsilon \in \{s', t'\}$, $w'/\epsilon \in \{s', t'\}$. By hypothesis there exists β a common descendant of s' and t' by R. If $v'/\epsilon = s'$ and $w'/\epsilon = t'$, then rewriting s' (resp., t') to β in v' (resp., w') allows to derive terms that are identical below ϵ. Using similar reductions at the other occurrences of U (note that they are independent) we obtain h as in the lemma. □

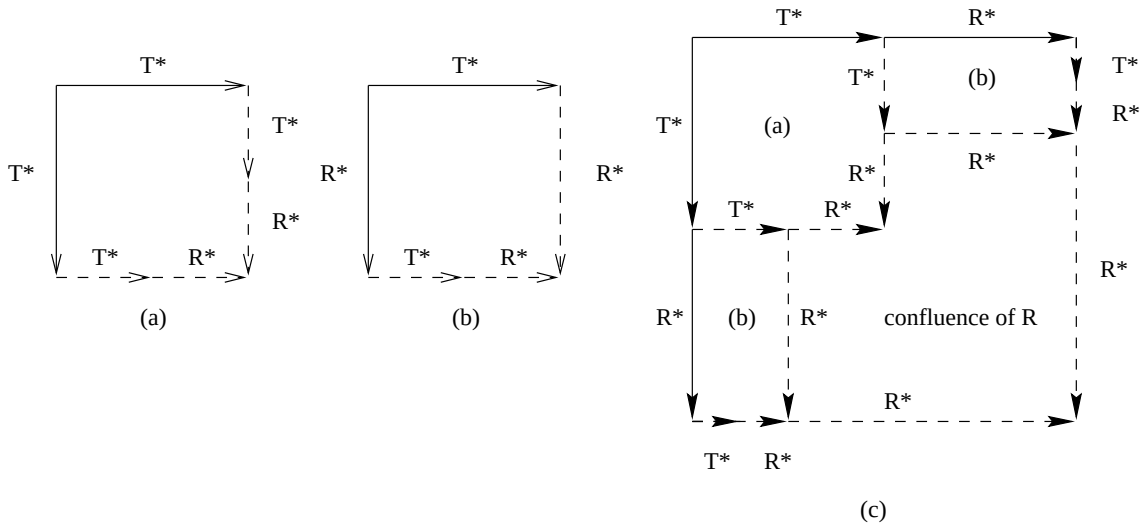

Figure 1:

Lemma 9 *If $u \xrightarrow{*}_T v$, $u \xrightarrow{*}_R w$ and v does not contain any occurrence of a then there exists w', h such that $v \xrightarrow{*}_R h$, $w \xrightarrow{*}_T w' \xrightarrow{*}_R h$, and there is no occurrence of a in w'.*

Proof: The lemma is illustrated by Figure 1 (b). Let U be the set of positions of a in u. Since v does not contain any a and the positions $u_1, \ldots, u_m$ in U are independent, we can build a derivation $v \xrightarrow{*}_R v' = u[a \parallel \beta]$, by rewriting all the s', t' in v occurring at a position $\epsilon \in U$ to β. By induction on the number n of R steps in $u \xrightarrow{*}_R w$ we can show that $v' \xrightarrow{*}_R w[a \parallel \beta]$. If $n = 0$ it is obvious. Otherwise $u \xrightarrow{n-1}_R p \xrightarrow{l \to r, \epsilon}_R w$. By induction hypothesis: $u[a \parallel \beta] \xrightarrow{*}_R p[a \parallel \beta]$. Since a behaves as a free constant for R, we have that $p \xrightarrow{l \to r, \epsilon}_R w$ implies $p[a \parallel \beta] \xrightarrow{l \to r, \epsilon}_R w[a \parallel \beta]$ and this finishes the induction step. We can take $w[a \parallel \beta]$ for h and $w[a \parallel s'] = w'$ since $w \xrightarrow{*}_T w' \xrightarrow{*}_R w[a \parallel \beta]$. □

Lemma 10 $\xrightarrow{*}_T \cdot \xrightarrow{*}_R$ *is strongly confluent.*

Proof: See Figure 1 (c). The upper left diagram is closed using Lemma 8. The upper right and lower left are closed using Lemma 9. The remaining part is closed using the confluence of R. □

Theorem 11 *Reachability $\leq_P$ Joinability.*

Proof: Given an instance R, s, t of Reachability we construct an instance R', s', t' of Joinability.
Case 1: R is ground. Let $R' = R \cup \{t \to c\}$, where c is a new constant not in R, s, t. Let $s' = s$ and $t' = c$. We show that $s \xrightarrow{*}_R t$ iff $s' \downarrow_{R'} t'$. The only if direction is obvious. Now

suppose that $s' \downarrow_{R'} t' = c$. Since c is a normal form, this implies $s' \stackrel{*}{\rightarrow}_{R'} c$. Since $t \rightarrow c$ is the only rule with c as the right-hand side, this implies $s' = s \stackrel{*}{\rightarrow}_{R'} t$ and by considering a shortest reduction from s to t in R' it is clear that $s \stackrel{*}{\rightarrow}_{R} t$.
Case 2: R is not ground. Without loss of generality we may assume that s is not a variable, since in this case t is never reachable from s. If t is a variable the rule for t is dropped from R' below. We may also assume that $(Var(s) \cup Var(t)) \cap Var(R) = \emptyset$. The variables of s and t become new constants of R'. Let $R' = R \cup \{t \rightarrow d\}$ where d is a new constant not in R, s, t. Let $s' = s$ and $t' = d$. The rest of the argument is similar to the ground case.

Note that R' is ground iff R is and that the reduction can be done in logarithmic space and hence also polynomial time. □

Theorem 12 *Joinability* $\leq_P$ *Reachability.*

Proof: Given an instance R, s, t of Joinability, we let $R' = R \cup \{equal(x, x) \rightarrow true\}$, where $equal$ and $true$ are new function symbols not in the signature of R, s, t. Let $s' = equal(s, t)$ and $t' = true$. It is easy to see that $s \downarrow_R t$ iff $s' \stackrel{*}{\rightarrow}_{R'} t'$. □

Thus, we have the following tight relationship between Reachability and Joinability for general rewrite systems.

Corollary 13 *Reachability* $\equiv_P$ *Joinability.*

Remarks. Note that in Theorem 11 the reduction preserves all properties of the given rewrite system, e.g., groundness, left-linearity, etc. However, in Theorem 12 this is not the case. We can change the reduction to preserve left-linearity by allowing variables in the right-hand sides, but whether groundness can be preserved is open.

Theorem 14 *Joinability for non-collapsing, variable preserving systems* $\leq_P$ *Not* $UN^{\rightarrow}$.

Proof: Given an instance R, s, t of Joinability we construct an instance R' of $UN^{\rightarrow}$ as follows.
Case 1: R is ground. Let $R' = R^{-1} \cup T \cup \{s \rightarrow c, t \rightarrow d\}$, where c, d are new constants not in R, s, t, $R^{-1} = \{r \rightarrow l \mid l \rightarrow r \in R\}$, and $T = \{a \rightarrow a \mid$ for every constant $a \in R, s, t\}$. Note that R' is also ground. We must show that $s \downarrow_R t$ iff R' is Not $UN^{\rightarrow}$. The only if direction is obvious since $R^{-1} \subseteq R'$. For the if direction, suppose that there is a term S such that $S \stackrel{*}{\rightarrow}_{R'} n$ and $S \stackrel{*}{\rightarrow}_{R'} N$ for distinct normal forms n, N. Since R' is ground, we may assume, without loss of generality, that S is also ground. We may further assume that S does not contain c or d, since R' cannot reduce any subterms containing c or d. Because of the T rules of R' we conclude that $\{n, N\} = \{c, d\}$. Therefore, we must have reduction sequences $S \stackrel{*}{\rightarrow}_{R'} s \rightarrow_{R'} c$ and $S \stackrel{*}{\rightarrow}_{R'} t \rightarrow_{R'} d$. By taking the shortest such reduction sequences we conclude that $S \stackrel{*}{\rightarrow}_{R^{-1}} s$ and $S \stackrel{*}{\rightarrow}_{R^{-1}} t$, which implies that $s \downarrow_R t$.
Case 2: R is not ground. Without loss of generality we may assume that $(Var(s) \cup Var(t)) \cap Var(R) = \emptyset$. The variables of s and t become new constants of R'. If either s or t is a variable, the corresponding rule is dropped from R' below.

Let $R' = \{r \rightarrow l \mid l \rightarrow r \in R\} \cup \{s \rightarrow c, t \rightarrow d\} \cup T'$, where c, d are new constants not in R, s, t. $T' = T \cup \{x \rightarrow x \mid x \in Var(s) \cup Var(t)\} \cup \{h(z_1, \ldots, z_p) \rightarrow h(z_1, \ldots, z_p) \mid$ for every

function symbol h of arity $p > 0$ in $R, s, t\}$. We must show that $s \downarrow t$ iff R' is Not $UN^{\rightarrow}$. The only if direction is obvious since $R^{-1} \subseteq R'$. For the if direction, suppose that there is a term S such that $S \stackrel{*}{\rightarrow}_{R'} n$ and $S \stackrel{*}{\rightarrow}_{R'} N$ for distinct normal forms n, N.

Because of the T' rules of R' we conclude that $\{n, N\} = \{c, d\}$. Therefore, we must have reduction sequences $S \stackrel{*}{\rightarrow}_{R'} s \rightarrow_{R'} c$ and $S \stackrel{*}{\rightarrow}_{R'} t \rightarrow_{R'} d$. By taking the shortest such reduction sequences we conclude that $S \stackrel{*}{\rightarrow}_{R^{-1}} s$ and $S \stackrel{*}{\rightarrow}_{R^{-1}} t$, which implies that $s \downarrow_R t$.

Finally, observe that in both cases the reduction can be done in polynomial time. □

It is not difficult to show that Joinability for confluent systems $\leq_P$ Normal form reachability for confluent systems (but the reduction does not preserve groundness) and Joinability for confluent systems $\leq_P$ WP (word problem). Now we show that Confluence is undecidable for a recursive family of string rewrite systems and that Word Problem is decidable for the same family.

Theorem 15 *There exists a family of string rewrite systems for which Confluence is undecidable and for which WP is trivially decidable.*

Proof: Let R be a Thue system on alphabet $\Sigma = \{0, 1\}$. Let R' be the system $T \cup S$, where $S = \{s \rightarrow t | s = t \text{ or } t = s \in R\}$ and $T = \{g \rightarrow 0g, g \rightarrow 1g, g \rightarrow \lambda\}$. Whether the monoid generated by R is trivial is known to be undecidable, since *triviality* is a Markov property.

Assume that R is nontrivial. Then there exists two words w_1, w_2 on Σ such that $w_1 \not\stackrel{*}{\leftrightarrow}_R w_2$. Since g generates Σ^*, we have $g \stackrel{*}{\rightarrow}_T w_1$ and $g \stackrel{*}{\rightarrow}_T w_2$. Since w_1 and w_2 do not have any g symbol they can be rewritten only by rules from S. But w_1, w_2 cannot be joinable since they are not congruent in R.

Assume that the monoid associated to R is trivial. Then note that every rewrite derivation $a \stackrel{*}{\rightarrow}_{R'} b$ can be extended to $b \stackrel{*}{\rightarrow}_T c$ where c is in Σ^* and $c \stackrel{*}{\rightarrow}_S \lambda$. Therefore every words rewrite to λ which implies that R' is confluent.

The Word Problem for R' is obviously decidable since every word is equal to g modulo R'. □

Linear E-Matching

Linear E-Matching.
Instance. A linear equational theory E, two terms $s \in T(\Sigma, V)$ and $t \in T(\Sigma)$ such that a variable has not more than one occurrence in s.
Problem. Is there a substitution σ such that $\sigma(s) =_E t$?

It is easily seen that $WP \leq_P$ E-matching. We now show:

Theorem 16 *There exists a linear equational theory E for which Linear E-Matching is undecidable and WP is decidable.*

Proof: Let $P = \{(v_i, w_i) | i = 1, \ldots n\}$ be a fixed set of pairs of strings on Σ. Let us consider the following variant of Modified Post Correspondance Problem from Narendran-Otto. Given two strings $v_0, w_0 \in \Sigma^*$ is there a sequence $i_1, \ldots, i_k$ of numbers with each i_j between 1 and n such that $v_{i_k} v_{i_{k-1}} \ldots v_{i_1} v_0 = w_{i_k} w_{i_{k-1}} \ldots w_{i_1} w_0$.

We consider the following rewrite system R_P whose signature contains n unary symbols $1, \ldots, n$, a unary symbol for each element of the alphabet Σ of P, a single constant $\perp$ and functions $k, check$:

$$\begin{array}{lllll} 1 & k(y, a(x), a(u), v, w) & \rightarrow & k(y, x, u, a(v), a(w)) & \text{for any } a \in \Sigma \\ 2 & k(y, \perp, \perp, v, w) & \rightarrow & check(y, v, w) & \\ 3 & check(i(y), v_i(x), w_i(u)) & \rightarrow & check(y, x, u) & \text{for any } (v_i, w_i) \in P \end{array}$$

Let E be the equational theory associated with R_P. Since the system R_P terminates and has no critical pairs it is Church-Rosser. Therefore WP is decidable. Let us consider the linear E-matching problem, where $s = k(y, x, u, \perp, \perp)$ and $t = check(\perp, v_0(\perp), w_0(\perp))$.

If there is a solution π to the matching problem, then $k(y, x, u, \perp, \perp)\pi$ rewrites to $check(\perp, v_0(\perp), w_0(\perp))$ since $check(\perp, v_0(\perp), w_0(\perp))$ is irreducible and the system R_P is Church-Rosser. In this rewriting sequence one needs to use rule 2. Therefore $x\pi = u\pi$ and $x\pi$ represents a word. Then the derivation can be extended to $check(\perp, v_0(\perp), w_0(\perp))$ only by applying rules 3 and this shows that $x\tilde{\pi}$ is a solution of P, where $\tilde{v}$ is the reverse of v. The converse is easy. □

3.1 Reductions for Rewriting modulo E

Note that most of these reductions work for rewriting modulo E for general E. However, for the purpose of this paper it is sufficient to observe that all of the reductions work for rewriting modulo E for variable-preserving, noncollapsing equational theories E. Variable-preserving, noncollapsing equational theories include associativity and commutativity. The proof of this observation is straightforward since R/E is equivalent to $R \cup \{s \rightarrow t \mid s = t \in E \vee t = s \in E\}$ for variable-preserving, noncollapsing E.

4 Rewriting Problems with Polynomial-time Algorithms

4.1 $UN^{=} \in P$ for ground systems

We now give a polynomial time algorithm for deciding the unicity property of ground rewrite systems. The algorithm makes use of the congruence closure algorithm [6, 9, 11] to keep the necessary (potentially infinite) set of equational proofs in a finite compact data structure. The data structure consists of a set of equivalence classes of terms known to be equivalent using the given rules, which are treated as equations. Terms are implicitly represented by signatures. Each class is assigned a number and the signature of a term $f(t_1, \ldots, t_n)$ is the $n+1$-tuple $\langle f, \#[t_1], \ldots, \#[t_n] \rangle$, where $\#[t_i]$ is the number of the equivalence class which contains the signature representing t_i. We say that an *equivalence class represents a term t* if it contains the signature representing t.

Our algorithm works in two phases. In the first phase the algorithm constructs the signatures of all the subterms appearing in the given rewrite system and uses the rules as equations to form the union of the equivalence classes containing signatures representing the

lhs and rhs of the rule. This union may cause other signatures to change and this may create further unions. After all the rules have been used the algorithm enters the second phase, which is described below. The polynomial time implementation of the following algorithm is discussed later.
Algorithm: UN(R) /* R is a ground rewrite system */

1. Construct the congruence-closure data structure using all the rules as equations.
Let C represent the set of all equivalence classes in the final data structure.
2(a). $H = 0$; $NF = T' = T = \emptyset$; $Type1class = Type2class = C$;
2(b). **repeat**
  Let S be the set of all signatures in C whose arguments are all in $Type2class$;
  Let T be the set of terms of height H whose arguments are
  terms all of which are represented by signatures in $Type2class$;
  Let T' be the members of T that are lhs's of R and let $NF = NF \cup T - T'$;
  $Type1class = Type1class \cap \{c \mid c \in C$ and c represents no member of $NF\}$;
  $Type2class = C - Type1class$;
  If $Type2class = \emptyset$ **return** (true) and **halt** ;
  For each $c \in Type2class$, let $T(c) = \{s \in c \mid s$ represents a term in $T\}$;
  If there is a $c \in Type2class$ with $|T(c) \cap NF| > 1$ **return**(false) and **halt** ;
  $H = H + 1$;
until $H = |R|$;
return(true) and **halt** ;

We now prove the correctness of this algorithm. For this we require the following proposition. We say that a symbol appears in R if it appears in the lhs or rhs of some rule in R. The *size* of a term is the number of function and variable symbols in it. The size of a rewrite system R, denoted $|R|$, is the sum of the sizes of all the lhs's and rhs's of R. The *height* of a term is the number of edges in a longest path from the root to a leaf in its tree representation.

Proposition 17 *Let R be any ground rewrite system. If there are two distinct normal forms t_1 and t_2 satisfying $t_1 =_R t_2$, then there are two ground normal forms t_1 and t_2 satisfying $t_1 =_R t_2$. Further the t_i's can be chosen so that they contain only function symbols appearing in the rules of R.*

Proof: We show by induction on the length of the equational proof $q : t_1 =_R t_2$ that if there is any symbol in any of the t_i's at occurrence o, which does not appear in R, then no reduction can be made at a prefix of o in q. The rest is straightforward using the projection property for equational proofs. □

We now prove the following lemmas which are crucial to the correctness of the algorithm. The first lemma allows us to restrict the search for equational proofs to only those that are present in the output of the congruence closure algorithm on the rewrite system R. However, since even this set can be "very large" (infinite in general) the next lemma allows us to restrict the search space even further to those that are between terms of "minimal" height.

Lemma 18 *Let R be any ground rewrite system. If there exist two distinct normal forms t_1 and t_2 such that $t_1 =_R t_2$, then there exist two distinct ground normal forms t_1 and t_2 such that $t_1 =_R t_2$ and both t_1 and t_2 are represented by signatures in the output of the congruence-closure algorithm on R.*

Proof: By Proposition 17, we can restrict ourselves to equational proofs between ground normal forms containing only the function symbols of R. Let s and t be any two ground terms such that $q : s =_R t$. Let q be $s = s_0 \leftrightarrow s_1 \ldots \leftrightarrow s_n = t$ for some $n \geq 0$. First, we show that if any one of the terms in the equational proof q is represented in the output of the congruence closure algorithm on R, then *every* term must be represented in the output of the congruence closure algorithm on R. Suppose that s_i is represented for some i, $0 \leq i \leq n$. We show that both s_{i-1} and s_{i+1} (if they exist) must also be represented.

Consider $s_{i-1} \leftrightarrow s_i$. If $s_i \rightarrow s_{i-1}$, then $s_i = C[S]$ for some context C and term S and $s_{i-1} = C[T]$ and $S \rightarrow T$ is a rule in R. Since we start with all the subterms of lhs's and rhs's represented and we never lose any represented terms by applying congruence closure, both S and T are represented. Since $S \rightarrow T$ both S and T are represented by the same class in the output of the congruence-closure algorithm (CCA) on R. Since $C[S] = s_i$ is represented, there are signatures corresponding to each superterm (in s_i) of S in the output of CCA and since S and T are represented by the same class and $s_{i-1} = C[T]$, the *same* signatures represent every superterm (in s_{i-1}) of T in $C[T]$ and hence s_{i-1} is also represented.

Now suppose that $s_{i-1} \rightarrow s_i$, then $s_{i-1} = C[S]$ for some context C and subterm S and $s_i = C[T]$ for some subterm T and $S \rightarrow T \in R$. Again we know that S and T are represented by the same class and we know that there exist signatures corresponding to every superterm (in s_i) of T since s_i is represented by assumption. Therefore, s_{i-1} must also be represented.

Finally, suppose there are two distinct ground normal forms with an equational proof $q : t_1 =_R t_2$ such that every term in q is not represented in the output of CCA on R. This implies that there is no root reduction in q, since a root reduction implies that the corresponding terms are represented and hence all terms are represented, as we have just now shown. Therefore, we consider the minimal occurrences at which reductions are applied in q (there must be at least one such occurrence; since t_1 and t_2 are distinct q cannot be a null proof) and use the projection property of equational proofs to get two distinct ground normal forms t_1' and t_2' which are proper subterms of t_1 and t_2 such that $q' : t_1' =_R t_2'$ and there is a root reduction in q'. The rest follows. □

Lemma 19 *Let R be any ground rewrite system. If there exist two distinct normal forms t_1 and t_2 such that $t_1 =_R t_2$, then there exist two distinct ground normal forms t_1 and t_2 such that $t_1 =_R t_2$ and the height of the t_i's is at most $|C| \leq |R|$.*

Proof: By Proposition 17 and Lemma 18, we can restrict ourselves to equational proofs between distinct ground normal forms containing only the function symbols of R and which are represented in the output of CCA on R. Suppose that the height of any one of the ground normal forms, say t_1, in the equational proof $q : t_1 =_R t_2$ is more than $|C|$, where both t_1 and t_2 are represented in the output of CCA.

Consider a longest path in t_1 from the root of t_1 to a leaf. Since this path has at least $|C|$ edges, there are at least $|C|+1$ nodes on this path including the root and the leaf. Hence there are at least $|C|+1$ distinct ground subterms of t_1 on this path. Since t_1 is a represented term, every subterm of t_1 is also represented by some class and since there are only $|C|$ equivalence classes, by the pigeonhole principle there must be two distinct ground subterms s and t on this path that are represented by the same equivalence class. By the property of the congruence-closure algorithm we have $s =_R t$. Since subterms of normal forms are normal forms (by definition of a normal form), s and t are ground normal forms. By choosing s and t of minimal height with these properties, the lemma is proved. □

Theorem 20 *Let R be any ground rewrite system. R has the unicity property iff Algorithm UN returns true.*

Proof: We prove that there exist distinct ground normal forms t_1 and t_2 of height h (i.e. at least one of them has height h and the other at most h) such that $t_1 =_R t_2$, iff after h iterations of the repeat loop: (a) Algorithm UN returns false, and (b) the classes in $Type2class$ together represent all normal forms of height at most h that can be constructed from function symbols appearing in R. The proof is by induction on the number of iterations of the repeat loop in Algorithm UN. By Lemma 19 we may restrict ourselves to distinct ground normal forms of height at most $|C|$.

Base case: Assume that there are distinct ground normal forms t_1 and t_2 of height 0 such that $t_1 =_R t_2$. This implies that t_1 and t_2 are distinct irreducible constants represented by the same class in C, say class c. After the first iteration of the repeat loop T contains all the constants of R and $NF = T - T'$ contains all the irreducible constants of R. Type1class contains all classes that contain no irreducible constants and $Type2class$ contains all classes that contain at least one irreducible constant. Since t_1 and t_2 are represented by some class in C, $Type2class \neq \emptyset$. In particular, there is a class $c \in Type2class$ representing t_1 and t_2. Hence $|T(c)| \geq 2$ and the Algorithm returns false and terminates. Moreover, the classes in $Type2class$ together represent all the irreducible constants appearing in R. It is easy to see that if Algorithm UN returns false after one iteration, then there are two distinct irreducible constants a and b represented by the same class and hence $a =_R b$ and R does not have the unicity property.

Induction Step: Assume that there are distinct ground normal forms t_1 and t_2 of height at most $h+1$ such that $t_1 =_R t_2$. If both t_1 and t_2 are of height at most h, then by the induction hypothesis Algorithm UN must return false after h iterations. Thus at least one of t_1 and t_2 must be of height exactly $h+1$. Also, by the induction hypothesis, every class in $Type2class$ after exactly h iterations of the repeat loop represents exactly one member of the set of normal forms of height at most h (since the Algorithm did not return false after h iterations). The set of normal forms of height $h+1$ must have at least one proper subterm of height h and the other proper subterms of height at most h. By the induction hypothesis we have that the classes in $Type2class$ together represent all the irreducible terms of height at most h. Therefore, in iteration $h+1$ of the repeat loop, T will contain both t_1 and t_2 since both must be represented by signatures whose arguments are classes in $Type2class$. Now

since both terms are normal forms, they cannot be lhs's of rules in R. Hence $NF = T - T'$ must contain both t_1 and t_2. Also, if c is the class that represents t_1 and t_2, then c must be in $Type2class$. In fact, every member of NF must be represented by some class in $Type2class$ by the construction. Also c cannot contain just a single signature σ representing both t_1 and t_2 since this means that some class in the argument list of σ represents two distinct normal forms, which would have caused termination in iteration h. Hence c contains two distinct signatures representing t_1 and t_2 and this causes the Algorithm to return false in the $(h+1)$th iteration. By the construction of the Algorithm it is clear that if the Algorithm returns false there must exist distinct ground normal forms t_1 and t_2 with $t_1 =_R t_2$.

This concludes the proof. □

Now we analyze the complexity of Algorithm UN and show that it runs in time polynomial in the input size $|R|$. In the analysis we use only simple data structures and somewhat loose arguments, since our main objective here is merely to establish that the algorithm takes time which is a polynomial in the input size. The main difficulty is to construct the set of normal forms represented in C efficiently. A naive procedure that generates all possible normal forms of height h using the function symbols appearing in R in a bottom-up manner and checks whether they are represented by some class in C may take exponential time.

Theorem 21 *The running time of Algorithm UN is a polynomial in the input size* $|R|$

Proof: Step 1 clearly takes polynomial time, which is proved by [6, 9, 11]. Note that number of signatures in the output of step 1 is bounded from above by $|R|$ and hence so is $|C|$. We assume that after step 1 we have a list of classes and each class itself is stored in the form of a list of signatures. Each signature is stored in a list with the root symbol followed by the arguments which are numbers of classes. A naive implementation of Step 2(a) takes at most $O(|R|)$ time.

The idea is to generate the normal forms of height h that are represented by some class in C in the hth iteration. As each normal form is generated its signature and then class is determined. The class is stored in a temporary list L and also a global list GL. Every time a class is to be inserted in L or GL we check whether it exists already in L or in GL and if so return false and terminate. Otherwise we keep the list L and GL and go to the next iteration, where L is used to find the next batch of normal forms and then initialized to null. GL is kept throughout the algorithm.

In detail, we implement the idea as follows. In iteration h, we first find all those signatures all of whose arguments are in $Type2class$ and for which at least one argument is a class in L. The signatures are collected together in a list LS and L is now set to null. Now for each signature σ in LS we find the normal form in each class (there is exactly one so it can be kept track of efficiently) that is an argument of σ. Thus, we are able to construct the terms of T. Now, we check for each term in T whether it is an lhs. If it is not, then it is a normal form of height h. We keep track of the class to which each signature in LS belongs, this way we know the class of all the new normal forms in $T - T'$. The classes of these new normal forms are inserted at the end of the list L and GL making sure that the class being inserted

is not already in L or in GL. Note that the sizes of all these lists is at most $|C|$. Hence, all these operations can be done in time polynomial in $|R|$. □

4.2 Other rewriting problems in P

As a consequence of our reductions and Algorithm UN, we painlessly derive:

Corollary 22 *There are polynomial-time algorithms for: Normal form reachability for confluent ground systems and joinability of confluent ground systems.*

Proof: Use the reductions of these to $UN^=$ and the polynomial-time algorithm for $UN^=$. □

Note that these problems are known to be decidable for arbitrary ground systems [12, 13, 3, 5] by proofs from scratch. However, no upper bounds are given for the decision procedures in these papers. It can be shown that the algorithm for reachability in [12] takes polynomial time.

4.3 Polynomial Algorithms for Rewriting Problems modulo Commutativity

The congruence-closure algorithm and the polynomial-time algorithm for $UN^=$ can be generalized to handle commutative operators. The idea is to sort signatures based on the ordering of class numbers. The signature of a term is constructed in a bottom-up manner and the two arguments of each commutative operator are sorted according to the class representing them. Thus, we also have polynomial-time algorithms for normal form reachability of confluent ground systems modulo commutativity and joinability of confluent ground systems modulo commutativity.

5 Rewriting Problems modulo Associativity

We show that all rewriting problems of Section 3 are undecidable for ground rewriting modulo associativity by showing that Normal form reachability for ground confluent systems modulo associativity is undecidable. The latter is undecidable by reduction from the halting problem of Turing Machines that halt with blank tape, as detailed below. We use a coding of TM as a rewrite system R similar to [1]. Let $M = (\Sigma_0, Q, q_0, \delta)$ be a fixed deterministic TM whose halting problem is undecidable, where Σ_0 is the tape alphabet, Q the finite set of states, q_0 the initial state, and $\delta : Q \times \Sigma_0 \to Q \times (\Sigma_0 \cup \{L, R\})$ the transition function. The blank symbol is denoted by s_0 and the left (resp. right) move is denoted by L (resp. R). Without loss of generality we may assume that $q_a \in Q$ is the unique accepting state and there are no transitions from this state. We consider now the following signature of constant function symbols for the rewrite system R: $\Sigma_0 \cup \Sigma_0' \cup Q \cup \{h, h', q\}$ where $\Sigma_0' = \{s_i' | s_i \in \Sigma_0\}$, $\Sigma_0 \cap \Sigma_0' = \emptyset$, and h, h', q are new symbols.

$q_i s_p \to q_j s_l$ **if** $(q_i, s_p, q_j, s_l) \in \delta$ $\quad$ $s'_l q_i s_p \to q_j s_l s_p$ **if** $(q_i, s_l, q_j, L) \in \delta$
$q_i h \to q_j s_l h$ **if** $(q_i, s_0, q_j, s_l) \in \delta$ $\quad$ $s'_l q_i h \to q_j s_l h$ **if** $(q_i, s_l, q_j, L) \in \delta$
$q_i s_p \to s'_p q_j$ **if** $(q_i, s_p, q_j, R) \in \delta$ $\quad$ $h' q_i s_p \to h' q_j s_0 s_p$ **if** $(q_i, s_p, q_j, L) \in \delta$
$q_i h \to s'_0 q_j h$ **if** $(q_i, s_0, q_j, R) \in \delta$ $\quad$ $h' q_i h \to h' q_j s_0 h$ **if** $(q_i, s_0, q_j, L) \in \delta$

The reachability problem for R is undecidable since the TM M halts on input string w with the tape blank iff $h' q_0 w h \stackrel{*}{\to} h' q_a h$. Note that $h' q_a h$ is a normal form. We deduce:

Theorem 5.1 *The following problems are all undecidable for fixed ground rewrite systems modulo associativity: Normal form reachability for confluent systems, Joinability for Confluent Systems, $UN^=$, Confluence, Reachability, Joinability, and $UN^\to$.*

Proof: Note that if the restricted version of a problem is undecidable, then the general version is also undecidable. Hence we only need to prove the first four undecidable because of the reduction to $UN^\to$. Also, because of the reductions among the first four problems we only need to show that the first is undecidable. Therefore, the only point we are left to prove is that R above is confluent. A variant of this system R is proved confluent in [1] (lemma 2.6) under the assumption that the TM is terminating. We now relax this assumption. Note that there no overlaps between left-hand sides. We decompose any word as subwords of type $w'qw$ and $w'w$ where w (resp. w') is a maximal Σ_0 (resp. Σ'_0) subword and $q \in Q$. We can notice that such a decomposition is unique. Let w be a word and $W_1.W_2 \ldots W_k$ its decomposition. If $w \stackrel{*}{\to} w'$ then w' can be decomposed as $W'_1.W'_2 \ldots W'_k$ with $W_i \stackrel{*}{\to} W'_i$ for $i = 1 \ldots k$. This is because rewriting can take place only in subwords of type $w'qw$. Now if $w \stackrel{*}{\to} w'$ and $w \stackrel{*}{\to} w''$, it implies that $W_i \stackrel{*}{\to} W'_i$ and $W_i \stackrel{*}{\to} W''_i$ for $i = 1 \ldots k$. Since the TM is deterministic either $W'_i \stackrel{*}{\to} W''_i$ or $W''_i \stackrel{*}{\to} W'_i$. Therefore since rewriting steps occurring in different components commute we have $w' \downarrow w''$. □

Note that [1] a family of rewrite systems modulo associativity is constructed and confluence is shown undecidable for this family whereas all the undecidability results of this section are for a fixed rewrite system modulo associativity.

6 Rewrite Problems in Commutative Semigroups

6.1 A lower bound on Confluence, $UN^=$ modulo AC

Word problem for Equational theories.
Instance: Equational theory E, terms s, t. **Question:** Does $s =_E t$?

Theorem 23 *Word problem for weakly variable preserving theories $\leq_P$ Not $UN^=$.*

Proof: Let $R = \{l \to r \mid l = r \in E, Var(r) \subseteq Var(l)\} \cup \{r \to l \mid l = r \in E, Var(l) \subseteq Var(r)\} \cup \{s \to c, t \to d\} \cup T'$, where c, d are new constants not in E, s, t and T' is as in the proof of Theorem 6. We must show that $s =_E t$ iff $c =_R d$, i.e., R in Not $UN^=$. This follows from lemma 5 and the proposition in the Appendix. □

Theorem 24 *Word problem for noncollapsing, variable-preserving equational theories $\leq_P$ Confluence.*

Proof: Given E, s, t we construct an instance R of Confluence by taking $R = \{l \to r \mid l = r \in E \vee r = l \in E\} \cup \{a \to s, a \to t\}$, where a is a new constant. □

We consider now the case where the signature of the rewrite system R is built solely from constants and a single AC symbol. These rewrite systems are also called *commutative semi-Thue systems*. They present commutative semigroups. It appears that the confluence problem with just one constant and one AC symbol is already harder than the word problem for this signature, since the confluence problem is not studied in [2] whereas the word problem with this restriction is shown decidable there.

Note that since the word problem is Exp-space hard for commutative semigroups [10], we have:

Corollary 25 *$UN^=$ and Confluence for commutative semi-Thue systems are Expspace-hard.*

6.2 Definitions and useful results

Terms can be considered here as commutative words on an alphabet: $A = \{a_1, \ldots, a_n\}$. Therefore each term can be written $a_1^{m_1} \ldots a_n^{m_n}$ and identified to a vector $(m_1, \ldots, m_n)$ of $\mathbb{N}^n$ where the m_i are nonnegative integers. From now on, we shall identify $a_1^{m_1} \ldots a_n^{m_n}$ and the vector $(m_1, \ldots, m_n)$. The vector addition is denoted by $+$ and n is called the *dimension* of the semigroup. A word s is greater than a word t, denoted by $s \succ t$ iff each component of s is greater than or equal to the corresponding component of t and at least one of them is strictly greater.

6.3 Decidability of confluence

We show how to encode the confluence problem into the home space problem for Petri nets which is decidable in our case. Let us recall that a *Petri net* is a tuple $(\Pi, \mathcal{T}, F)$ where $\Pi = \{p_1, \ldots, p_n)$ is a finite set of *places*, $T = \{t_1, \ldots, t_p\}$ is a finite set of *transitions*, $F : \Pi \times \mathcal{T} \cup \mathcal{T} \times \Pi \to \mathbb{N}$ is the flow function. A marking : $\Pi \to \mathbb{N}$ attachs non-negative integers to places. Given a marking M, the *transition* τ is enabled if $M(p) \geq F(p, \tau)$ for all $p \in \Pi$ and firing τ yields a new marking M' such that $M'(p) = M(p) - F(p, \tau) + F(\tau, p)$. A marking M' is *reachable* from a another marking M if it can be obtained by a sequence of transitions. Given an initial marking M_0, the set of markings reachable from M_0 is the reachability set of M_0 called $Reach(M_0)$ The *home space problem* for Petri net is the following one.

> *Instance: a Petri net $(\Pi, \mathcal{T}, F)$, an initial marking M_0, E of set of markings.*
> *Question: from every marking $M \in Reach(M_0)$, can we reach a marking of E?*

This problem is decidable for E a linear set of $\mathbb{N}^{|\Pi|}$ [EJ89]. The following Petri net P is used to encode the confluence problem for a rewrite system $l_i \to r_i$ for $i = 1, \ldots, p$ and $l_i, r_i \in \mathbb{N}^n$ into a home space problem for P.

- The set of places $\Pi = \{p_1, \ldots, p_n, p'_1, \ldots, p'_n, q_0, q_1\}$ has $2n+2$ places, the $2n$ first ones are used to represent a pair of terms of $\mathbb{N}^n$ and the last two ones are used to separate different stages of the computation. Actually the two last places can be seen as states since their values will always be 0 or 1, with $q_i = 1$ (resp. 0) indicating that we are (resp. are not) in state q_i.
- The initial marking is $M_0 = (\underbrace{0 \ldots, 0}_{2n}, 1, 0)$
- We have several group of transitions. For each group of transition we indicate why they are introduced. If unspecified the value of $W(p, \tau)$ or $W(\tau, p)$ is 0.
  - Compute a marking $(s_1, \ldots, s_n, s_1, \ldots, s_n, 1, 0)$ from M_0 while staying in state q_0.

    $\mathcal{T}$ contains the transition τ_i^1 for $i = 1, \ldots, n$. These transitions add 1 at component i and $i+n$, i.e. $W(\tau_i^1, q_0) = W(\tau_{i+n}^1, q_0) = 1$.
  - Stop the previous computation and go to state q_1.

    $\mathcal{T}$ contains the transition τ_{q_0,q_1} with $W(q_0, \tau_{q_0,q_1}) = W(\tau_{q_0,q_1}, q_1) = 1$.
  - Mimic rewrite rule on the first term (components $1, \ldots, n$) and stay in state q_1.

    For each rule $l \to r$ with $l = (\alpha_1, \ldots, \alpha_n)$ and $r = (\beta_1, \ldots, \beta_n)$ add the transition τ_r^1 where $W(\tau_r^1, p_i) = \alpha_i$ and $W(p_i, \tau_r^1) = \beta_i$ for $i = 1, \ldots, n$, and $W(\tau_r^1, q_1) = W(q_1, \tau_r^1) = 1$.
  - Mimic rewrite rule on the second term (components $n+1, \ldots, 2n$) and stay in state q_1.
    Define transitions τ_r^2 similarly as above (replace p_i by p'_i).

We shall identify a marking M and the vector $(M(p_1), \ldots, M(p_n), M(p'_1), \ldots, M(p'_n), M(q_0), M(q_1))$. In the same way if s is a n-uple, we write $(s, s, \ldots)$ for $(s_1, \ldots, s_n, s_1, \ldots, s_n, \ldots)$.

Proposition 26 *The marking $(t_1, t_2, 0, 1)$ is reachable from M_0 iff there is some s such that $s \xrightarrow{*} t_1$ and $s \xrightarrow{*} t_2$.*

Proof: It is sufficient to realize that the following facts are true.

- Fact 1. From $(0, \ldots, 0, 1, 0)$ we necessarily reach markings of the form $(s, s, 1, 0)$. Conversely any marking of this form can be reached from $(0, \ldots, 0, 1, 0)$. This is proved by an easy induction on the number of transitions for the first direction, and on the size of the marking for the other direction.
- Fact2. Any marking $(s, s, 0, 1)$ can be reached from $(0, \ldots, 0, 1, 0)$. Use the previous point and use the second transition.

- Fact3. Any sequence of rewriting $s \xrightarrow{*} s'$ can be simulated by rules of the third group of rules, therefore for any t_1, t_2 such that $s \xrightarrow{*} t_1$ and $s \xrightarrow{*} t_2$, we can reach $(t_1, t_2, 0, 1)$ from $(s, s, 0, 1)$. The proof is by induction on the number of application of rules. Conversely firing a transition amounts to do some rewriting on the term represented by the component $1, \ldots, n$ or $n+1, \ldots, 2n$.

□

Now we state the main proposition which yields the decidability of confluence.

Proposition 27 *The rewrite system is confluent iff* $E = \{(z_1, \ldots, z_n, z_1, \ldots, z_n, 0, 1) \mid z_1, \ldots, z_n \in \mathbb{N}\}$ *is a home space for* P.

Proof: The proof relies on the previous proposition.

- Necessary condition. We assume the confluence of the rewrite system. Let M be some marking reachable from M_0. We show that E can be reached from M.
  - Either the value of place $2n+1^{th}$ is 1 therefore $M = (m_1, m_1, 1, 0)$ (use fact 1). The transition τ_{q_0,q_1} is applicable yielding $M' = (m_1, m_1, 0, 1) \in E$.
  - Or the value of place $2n+1^{th}$ is 0. This implies that $M = (t_1, t_2, 0, 1)$. Such marking can be reached only in the following way: reach some $(s, s, 1, 0)$ then reach $(s, s, 0, 1)$ the reach $t_1, t_2, 0, 1)$ where $s \xrightarrow{*} t_1$ and $s \xrightarrow{*} t_2$ are rewrite sequences. Since the rewrite system is confluent there exists t such that $t_1 \xrightarrow{*} t$ and $t_2 \xrightarrow{*} t$. Using previous propositions, we have that $(t, t, 0, 1) \in E$ is reachable from M

  The set E can be reached from any marking reachable from M_0, therefore it is a home space.

- Sufficient condition. We assume that E is a home space. Let s, t_1, t_2 be any terms such that $s \xrightarrow{*} t_1$ and $s \xrightarrow{*} t_2$. We know that the marking $M = (t_1, t_2, 0, 1)$ is reachable from M_0. Since E is a home state, there is some $(z, z, 0, 1)$ reachable from M, i.e. there z such that $t_1 \xrightarrow{*} z$ and $t_2 \xrightarrow{*} z$. This proves the confluence of the rewrite system.

□

Since the set E is a linear set, the home space property for E is decidable [EJ89] and we get the theorem.

Theorem 28 *The confluence property for rewrite system in commutative semigroup is decidable.*

It is easy to see that we need only the decidability of the home space property for $E' = \{(0, \ldots, 0, 1)\}$ (substract the period of E). The proof of decidability of the home-state property relies heavily on the reachability property in Petri nets. The complexity of this problem is high, but no exact bound is known in the general case. Therefore, we don't have any precise informations on the complexity of our decision method, except in special cases.

7 Conclusion

In this paper, we have given polynomial-time reductions for some basic decision problems of rewrite systems, and we have given a polynomial-time algorithm for the Unique-normal-form problem (and problems reducible to it) of ground rewriting modulo commutativity. We have also shown the undecidability of several rewrite problems for ground rewriting modulo associativity. Finally, we have shown the decidability of the confluence problem for ground rewriting modulo an AC operator and constants. There are many directions for future work arising from this study. For example, more algorithms and polynomial-time reductions for the problems studied here and elsewhere are of considerable importance.

References

[1] G. Bauer and F. Otto. Finite complete rewriting systems and the complexity of the word problem. *Acta Informatica*, 21:521–540, 1984.

[2] R. Book and F. Otto. String Rewriting Systems *Springer-Verlag*, 1992.

[3] 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.

[4] N. Dershowitz and J.P. Jouannaud. Rewrite systems. In *Handbook of Theoretical Computer Science*, volume 2, chapter 6. North-Holland, 1990.

[5] A. Deruyver and R. Gilleron. The reachability problem for ground TRS and some extensions. In *Lecture Notes in Computer Science*, volume 351, pages 227–243, 1989.

[6] P.J. Downey, R. Sethi, and R.E. Tarjan. Variations on the common subexpression problem. *Journal of the ACM*, 27(4):758–771, 1980.

[EJ89] D. Frutos Escrig and C. Johnen. Decidability of home space property. Technical Report 503, Laboratoire de Recherche en Informatique, Université de Paris Sud, Bat 490, 91405 ORSAY, FRANCE, July 1989.

[7] J. Hopcroft and J.J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8:135–159, 1979.

[8] J.W. Klop. Rewrite systems. In *Handbook of Logic in Computer Science*. Oxford, 1992.

[9] D. Kozen. Complexity of finitely presented algebras. In *Proc. Ninth ACM Symposium on Theory of Computing*, pages 164–177, 1977.

[10] E. W. Mayr and A. R. Meyer. The complexity of the word problem for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46:305–329, 1982.

[11] G. Nelson and D.C. Oppen. Fast decision algorithms based on congruence closure. *Journal of the ACM*, 27:356–364, 1980. Also in the 18th IEEE FOCS, 1977.

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

[13] M. Oyamaguchi. The reachability and joinability problems for right-ground term rewriting systems. *Journal of Information Processing*, 13(3):347–354, 1990.

8 Appendix

Proposition 29 *Let $R' = R \cup S$, where $S = \{s \to c, t \to d\}$ and c, d are new constants not in R, s, t and s, t are ground terms. If $c =_{R'} d$ then $s =_R t$.*

Proof: An equational proof $c =_{R'} d$ can be decomposed as $c =_I A_1 =_J B_1 =_I \ldots = A_n =_J B_n =_I d$ or $c =_I A_1 =_J B_1 =_I \ldots = A_n =_I B_n =_J d$, where $I = S$ and $J = R$ or vice-versa. Wlog, we assume $c =_S A_1 =_R B_1 =_S \ldots = A_n =_S B_n =_R d$. Since c, d do not occur in R, using lemma 5 we have $A_{i+1}[c \parallel s, d \parallel t] =_R B_{i+1}[c \parallel s, d \parallel t]$. Since proofsteps between B_i and A_{i+1} amounts to replace c or d by s or t respectively and since c, d do not occur in s, t, we have also $B_i[c \parallel s, d \parallel t] = A_{i+1}[c \parallel s, d \parallel t]$. Doing the parallel replacements for all A_i and B_i gives a proof of the required type. □

Unit´e de recherche INRIA Lorraine, Technopˆole de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhˆone-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399