On the ground reducibility problem for word rewriting systems with variables

Gregory Kucherov, Michaël Rusinowitch

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

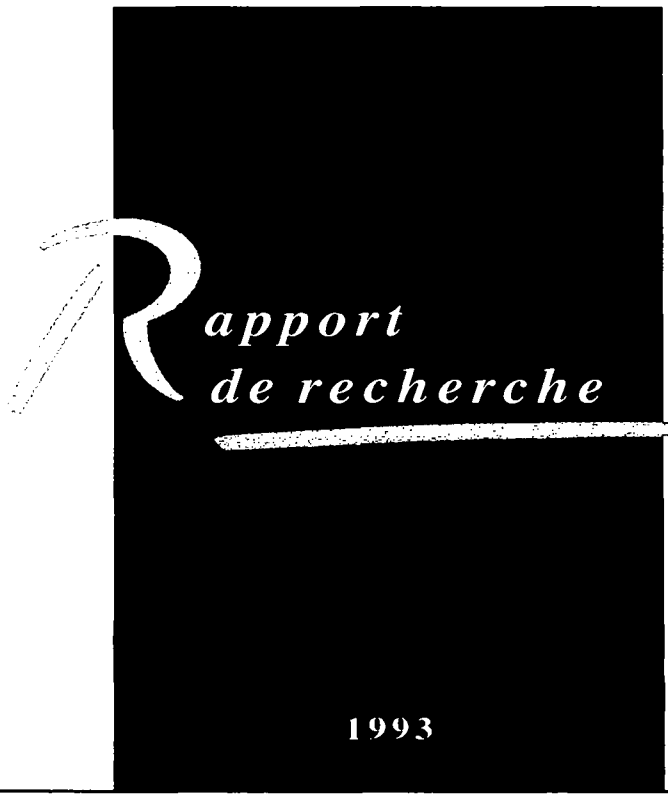
On the ground reducibility problem for word rewriting systems with variables

Gregory KUCHEROV
Michaël RUSINOWITCH

N° 1892
Mai 1993

PROGRAMME 2

Calcul Symbolique,
Programmation
et Génie logiciel



On the ground reducibility problem for word rewriting systems with variables

Sur la réductibilité inductive pour les systèmes de réécriture de mots avec variables

Gregory Kucherov Michaël Rusinowitch

Abstract

Given a word rewriting system with variables R and a word with variables w the question we are interested in is whether all the instances of w obtained by substituting its variables by non-empty words are reducible by R. We prove this problem to be generally undecidable even for a very simple word w, namely axa where a is a letter and x a variable. When R is left-linear, the question is decidable for arbitrary (linear or non-linear) w.

Résumé

Nous montrons que la propriété de réductibilité inductive est indécidable pour les systèmes de réécriture de mots avec variables en général et décidable pour les systèmes linéaires gauches.

On ground reducibility problem for word rewriting systems with variables

Gregory KUCHEROV
Michaël RUSINOWITCH
CRIN and INRIA-Lorraine
Campus Scientifique, BP 239
F-54506 Vandœuvre-lès-Nancy, France
email: {kucherov,rusi}@loria.fr

1 Introduction

This work is motivated by our interest in *word rewriting systems with variables (WRSV)*. A WRSV $\mathcal{R}$ is a collection of rewrite rules $w \to v$ where w, v are words over a finite alphabet A and a variable alphabet $\mathcal{X}$. From the viewpoint of general rewriting systems theory WRSV are rewriting systems over a signature consisting of a single binary associative function (concatenation) and finitely many constant symbols (letters). Although rewriting systems have been intensively investigated for already more than ten years, WRSVs have not received much attention so far.

If the rules of $\mathcal{R}$ do not contain variables, we obtain *word rewriting system* (string rewriting system, semi-Thue systems) that have been investigated for a long time [2]. In particular, it is well known that these systems are computational universal.

The difficulty in applying to WRSVs the methods developed in the term rewriting system theory such as the Knuth-Bendix completion algorithm lies in particular in the complexity of the unification problem for words with variables. Though this famous problem has been shown decidable by Makanin in 1978 [8] the decision algorithm is rather involved.

In this paper we are interested in the *reductive power* of WRSVs, that is in properties of the domain of the rewriting relation generated by different systems. More precisely, if $\mathcal{R}$ is a WRSV over an alphabet A, we are interested in the properties of the set $Red(\mathcal{R})$ of words in A^* reducible by $\mathcal{R}$. The questions we ask about $Red(\mathcal{R})$ are following:

- For a given word with variables w, are all the instances of w reducible by $\mathcal{R}$?
- Is $Red(\mathcal{R})$ co-finite, i.e. is the set of irreducible words finite?
- If not, is this set a regular language?

The three questions have been proved decidable recently for ordinary term rewriting systems. The first one is the well-known *ground reducibility* problem that was shown to be decidable in [11, 6]. The second problem was proved decidable for ordinary term rewriting systems in [6, 7]. Finally, for the last question the decidability was shown in [7, 4, 13].

However, in the presence of associative functions these problems turned out to be more complex. A ground reducibility problem was shown in [5] to be undecidable for the signature containing among other functions an associative function. This result, however, cannot be generalized to WRSVs, as the auxiliary non-associative functions are essential in the proof. The second question is also very difficult. We just mention that in [1] the decidability of this problem is cited open for a very restricted case: $\mathcal{R}$ consists of a single rule whose left-hand side does not contain constants.

In fact, as far as our presentation is only concerned with the reducibility of words with respect to a given WRSV, we can simply exclude right-hand sides from consideration. Thus, the questions above can be directly stated in the setting of (non-linear) pattern matching. (Here "pattern matching" is to be understood in the extended sense: a pattern w matches a word v iff v has a subword which is an instance of w). Thus, the first question could be rephrased as follows: given a set of patterns $\mathcal{R}$, is every string matched by a given pattern w also matched by some pattern from $\mathcal{R}$?

In this paper we prove this question to be generally undecidable. It remains undecidable even for a very simple word w, namely for $w = axa$ where a is a letter and x a variable. We show that all the questions above are decidable if $\mathcal{R}$ is linear, i.e. when every pattern of $\mathcal{R}$ does not contain multiple occurences of a variable. Note that for a linear $\mathcal{R}$, the ground reducibility problem will be shown decidable for arbitrary (linear or non-linear) w.

2 Preliminaries

Given a finite alphabet A and a (potentially infinite) alphabet $\mathcal{X}$, we consider words over $A \cup \mathcal{X}$ called *words with variables* (or *patterns*) and words over A called simply *words* (or *strings*). Obviously, the same variable may occur several times in a pattern. A substitution σ is a mapping from $\mathcal{X}$ to A^+.[1] It can be naturally extended to a mapping from $(\mathcal{X} \cup A)^+$ to A^+ (denoted by σ too) such that $\sigma(v) = w$ iff w is derived from v by replacing each occurence of a variable x in v by $\sigma(x)$. In algebraic terminology, σ is a homomorphism from $(\mathcal{X} \cup A)^+$ to A^+ regarded as free semigroups over $A \cup \mathcal{X}$ and A respectively, such that $\sigma(a) = a$ for every $a \in A$.

We use the usual operations on strings and string sets (concatenation, star, plus,...). Concatenation will be denoted either by $\cdot$ or just by putting down the arguments one after another. Let ε denote the empty string and $|w|$ the length of a word (with variables) w. A position of a symbol (a letter or a variable) in a pattern w is a nonnegative integer in $\{0, ..., |w| - 1\}$. If p_1, p_2 are positions in w and $p_1 < p_2$, then by $w[p_1 \leftarrow p_2]$ we denote the pattern obtained by deleting all symbols at the positions $\{p_1, \ldots, p_2 - 1\}$.

Given a set $\mathcal{R}$ of words with variables, $Inst(\mathcal{R})$ denotes the set of instances of patterns from $\mathcal{R}$, that is the set $\{\sigma(w) | w \in \mathcal{R}\}$ for all possible substitutions σ. We define $Red(\mathcal{R}) = \{u_1 \cdot v \cdot u_2 | v \in Inst(\mathcal{R}),\ u_1, u_2 \in A^*\}$, and $NF(\mathcal{R}) = A^+ \setminus Red(\mathcal{R})$. Our notation is motivated by term rewriting system vocabulary. Think of $Red(\mathcal{R})$ as the set of strings *reducible* by a WRSV (see introduction) with $\mathcal{R}$ being the set of left-hand sides of the rules. For this reason we sometimes also call elements of $\mathcal{R}$ *rules* without adding the prefix "the left-hand side of". Similarly, $NF(\mathcal{R})$ stands for the set of all *normal forms* (=

[1] Having in mind term rewriting system applications, we do not allow a variable to be substituted be the empty string. This assumption is technical and does not affect the soundness of the results.

irreducible words) for $\mathcal{R}$. A word with variables w is called *ground reducible* w.r.t. $\mathcal{R}$ iff $Inst(\{w\}) \subseteq Red(\mathcal{R})$.

3 Linear Case

In this section we show using regular languages techniques that if a WRSV $\mathcal{R}$ only contains linear patterns, the ground reducibility problem w.r.t. $\mathcal{R}$ is decidable for any word w with variables.

The first easy observation is that for a linear $\mathcal{R}$, $Red(\mathcal{R})$ is a regular language. Assume that $v = u_0x_1u_1x_2 \dots x_nu_n$, where $u_i \in A^*$, $0 \leq i \leq n$ and $x_i \in \mathcal{X}$, $1 \leq i \leq n$. Assume also that v is linear which means that variables $x_1, \dots, x_n$ are distinct. Clearly, the set $Inst(\{v\})$ is then regular, since it can be represented by the regular expression $u_0A^+u_1A^+ \dots A^+u_n$. Also, the set $Red(\{v\})$ is regular, as it is represented by the regular expression $A^*u_0A^+u_1A^+ \dots A^+u_nA^*$. Finally, since for $\mathcal{R} = \{v_1, \dots, v_m\}$ we have $Red(\mathcal{R}) = \bigcup_{i=1}^{n} Red(\{v_i\})$, we obtain the following

Proposition 1 *For a linear WRSV $\mathcal{R}$, the language $Red(\mathcal{R})$ is regular.*

Of course, since $NF(\mathcal{R}) = A^+ \setminus Red(\mathcal{R})$, the set of normal forms is regular too. In particular, we immediately obtain

Proposition 2 *For a linear WRSV $\mathcal{R}$, it is decidable if the set of normal forms $NF(\mathcal{R})$ is finite.*

Proof: It is well known that for regular languages finiteness is a decidable property. □

Before turning to the ground reducibility problem we point out another interesting property that is easily decidable for linear WRSVs. We say that WRSVs $\mathcal{R}_1$ and $\mathcal{R}_2$ are *equivalent* if $Red(\mathcal{R}_1) = Red(\mathcal{R}_2)$. Obviously, for linear WRSVs equivalence is decidable too. An interesting question is whether for a given WRSV $\mathcal{R}$ there exists an equivalent ground WRSV $\mathcal{G}$?

Proposition 3 *For a linear WRSV $\mathcal{R}$, it is decidable if there exists a set of words $\mathcal{G}$ such that $Red(\mathcal{R}) = Red(\mathcal{G})$.*

Proof: First we note that words from $\mathcal{G}$ should be themselves reducible by $\mathcal{R}$. Moreover, $\mathcal{G}$ can always be chosen such that for each $w \in \mathcal{G}$, $w \in Inst(\mathcal{R})$ but no proper subword of w belongs to $Inst(\mathcal{R})$. It is easy to see that the converse holds: a finite $\mathcal{G}$ exists if the set of w as above is finite. Since $Inst(\mathcal{R})$, $NF(\mathcal{R})$ are regular, the set of words w with this property is described by the regular expression $Inst(\mathcal{R}) \cap A \cdot NF(\mathcal{R}) \cap NF(\mathcal{R}) \cdot A$ where $\cdot$ denotes concatenation. Thus, a finite $\mathcal{G}$ equivalent to $\mathcal{R}$ exists iff the language defined by this expression is finite, which is decidable. □

Now we prove that for linear WRSVs the ground reducibility problem remains decidable for an arbitrary (non-linear) word with variables w to be tested. We need a notation to identify positions in $\sigma(w)$. Assume that $w = u_0x_1u_1x_2 \dots x_nu_n$ (where variables x_i may be equal). If p_i is the position of x_i then by $\sigma(p_i)$ we denote the position in $\sigma(w)$ which correspond to the beginning of the substring corresponding to x_i. Formally, we define recurrently $\sigma(p_1) = p_1$, and $\sigma(p_{i+1}) = \sigma(p_i) + |\sigma(x_i)| + |u_i|$ for $2 \leq i \leq n$.

Lemma 1 *Given a linear WRSV $\mathcal{R}$ and an arbitrary word with variables w, it is decidable if w is ground reducible w.r.t. $\mathcal{R}$.*

Proof: The idea of the proof is somewhat similar to that for ordinary term rewriting systems [6, 7]. We show that a constant $C(\mathcal{R}, w)$ can be computed such that if w is not ground reducible w.r.t. $\mathcal{R}$, then there exists a $\mathcal{R}$-irreducible instance $\sigma(w)$, and $|\sigma(x)| \leq C(\mathcal{R}, w)$ for each variable x in w. Below we prove that if for some variable x in w, $|\sigma(x)|$ exceeds the bound, then we can always modify $\sigma(x)$ by reducing its length and preserving the irreducibility of $\sigma(w)$.

Recall that $NF(\mathcal{R})$ is a regular language and assume that an automaton $\mathcal{A}$ recognizing $NF(\mathcal{R})$ has K states denoted $q_1, \ldots, q_K$. To each position p in a word $u \in NF(\mathcal{R})$ the automaton associates a state $\mathcal{A}(u, p) = q_i$ for some i, $1 \leq i \leq K$. We will use the usual pumping lemma trick: if p_1, p_2 are position in u, $p_1 < p_2$ and $\mathcal{A}(u, p_1) = \mathcal{A}(u, p_2)$, then $u[p_1 \leftarrow p_2]$ belongs also to the same language and thus is not $\mathcal{R}$-reducible.

We show now that $C(\mathcal{R}, w)$ can be set to be equal K^n where n is the maximal number of occurences of a variable in w. Assume that $\sigma(w)$ is not $\mathcal{R}$-reducible and suppose $|\sigma(x)| > C(\mathcal{R}, w)$ where x is a variable in w. Assume that x occurs at positions $p_1, \ldots, p_m$ in w ($m \leq n$). The idea is to find two distinct positions p', p'' in $\sigma(w)$ such that $\sigma'(w)$ still belongs to $NF(\mathcal{R})$ where σ' is the substitution defined by $\sigma'(x) = \sigma(x)[p' \leftarrow p'']$, and $\sigma'(y) = \sigma(y)$ if $y \neq x$. To do this, we have to prove that there exist positions p', p'' in $\sigma(w)$ satisfying the following property: for every j, $1 \leq j \leq m$, $\mathcal{A}(w, \sigma(p_j) + p') = \mathcal{A}(w, \sigma(p_j) + p'')$. Note that with every position p in $\sigma(x)$ we can associate a m-tuple of states $< \mathcal{A}(w, \sigma(p_1) + p), \ldots, \mathcal{A}(w, \sigma(p_m) + p) >$. It is clear that there are at most K^m different tuples of this form. Since p has at least $K^n + 1$ possible values where $n \geq m$, by pigeon hole principle we conclude that positions p', p'' with the desired property must exist. □

4 General Case

In this section we prove the main result: we show that the ground reducibility problem for WRSVs is generally undecidable. In our proof we will use the formalism of *Minsky machines* that is defined as follows. A machine $\mathcal{M}$ is operated by a *program* $\mathcal{P}$ which is defined to be a finite list of *instructions*. The instructions are supposed to be labeled by natural numbers from 1 to some L. The machine operates on two *counters* S_1, S_2 each containing a nonnegative integer. An instruction l is of one of the following three forms that have a transparent semantics (here $j \in \{1, 2\}$ and $l, l', l'' \in \{1, \ldots, L\}$):

(i) l : `ADD 1 TO` S_j; `GOTO` l';

(ii) l : `IF` $S_j \neq 0$ `THEN SUBTRACT 1 FROM` S_j; `GOTO` l' `ELSE GOTO` l'';

(iii) l : `STOP`;

Without loss of generality we will assume that in every program $\mathcal{P}$ there is only one instruction of type (iii) which occurs at the end of the program, i.e. its label is the total number of instructions in the program. The machine starts with executing instruction 1 and works until the command `STOP` is encountered. Note that the execution process is deterministic and has no failure situations. Thus, the execution of a program either ends up with the `STOP` command or lasts forever. It is known [10] that every partial recursive function on natural numbers can be computed by such a machine in the following sense:

Theorem 1 (Minsky [10]) *For every partial recursive function f on natural numbers there exists a program $\mathcal{P}$ that applied to the initial counter values $S_1 = 2^d$ and $S_2 = 0$, halts with the counter values $S_1 = 2^{f(d)}$ and $S_2 = 0$ if $f(d)$ is defined, and does not halt otherwise.*

We will use theorem 1 in the proof of our main theorem below. From now on we will always consider programs $\mathcal{P}$ that correspond to some partial recursive function in the sense of theorem 1 and we will assume that if, started with counter values $S_1 = 2^d$, $S_2 = 0$, the machine halts, then the final value of S_2 is 0. Theorem 1 implies that it is not generally decidable if for a given d, a program $\mathcal{P}$ terminates on the initial counter values $S_1 = 2^d$, $S_2 = 0$.

Our goal is to prove the following main theorem.

Theorem 2 *It is undecidable if a linear term is ground reducible by a given WRSV $\mathcal{R}$.*

Proof: Consider a partial recursive function and assume that $\mathcal{P}$ is a program for the Minsky machine that computes this function in the sense of theorem 1. The run of $\mathcal{P}$ on an input 2^d can be represented as a sequence of *configurations*

$$(1, 2^d, 0).(l_1, s_1^1, s_2^1), \ldots, (l_k, s_1^k, s_2^k), \ldots$$

where a configuration (l_k, s_1^k, s_2^k) means that the machine is about to execute instruction l_k, and s_1^k, s_2^k are the current values of the counters. If the execution of $\mathcal{P}$ terminates, the sequence is finite and the final configuration is $(L, 2^m, 0)$ for some $m \geq 0$.

Consider now the alphabet $A = \{*, a, b, \models, \#\}$ and let us encode a run of the machine by a (finite or infinite) word over A in the following way. A configuration (l_k, s_1^k, s_2^k) is encoded by the word

$$\underbrace{aa\ldots a}_{s_1^k+1}\underbrace{*\,*\ldots*}_{l_k}\underbrace{bb\ldots b}_{s_2^k+1}$$

and the whole run is represented by a sequence of such words separated by #. The first configuration is preceded by the symbols $\models\#$. If the run is finite, the final configuration is followed by $\#\models$ and the coding word has then the following form:

$$\models\#\underbrace{a\ldots a}_{2^d+1}*b\#\ldots\#\underbrace{a\ldots a}_{s_1^k+1}\underbrace{*\ldots*}_{l_k}\underbrace{b\ldots b}_{s_2^k+1}\#\ldots\#\underbrace{a\ldots a}_{2^m+1}\underbrace{*\ldots*}_{L}b\#\models$$

Let us now give the idea of the proof. Let w be the word $\models u\models$ where u is a variable and let $\sigma(w)$ be an instance of w. We construct a set $\mathcal{R}$ of patterns that depend on the program $\mathcal{P}$ and on $d \geq 0$ such that if a pattern v applies to (=matches a subword of) $\sigma(w)$, then $\sigma(w)$ is not a correct encoding of a (finite) execution of $\mathcal{P}$ on $S_1 = 2^d$, $S_2 = 0$. Moreover, we prove that *every* instance $\sigma(w)$ that does not represent a correct finite execution is reducible by some of the patterns. Thus, the system $\mathcal{R}$ will be constructed so that $\sigma(w)$ is irreducible w.r.t. $\mathcal{R}$ if and only if $\sigma(w)$ is the encoding of the finite execution of $\mathcal{P}$ on $S_1 = 2^d$, $S_2 = 0$ for some $d \geq 0$. Hence, the set of irreducible instances of w is non-empty iff this set consists of a single instance that represents the finite execution of $\mathcal{P}$ on $S_1 = 2^d$, $S_2 = 0$. Since it is not decidable if the execution of $\mathcal{P}$ on $S_1 = 2^d$, $S_2 = 0$ is finite or not, we conclude that ground reducibility is not a decidable property.

The patterns of $\mathcal{R}$ can be divided into two categories. The first one (groups 1-5) contains patterns without variables. These patterns apply to the instances of w that are

not meaningful for purely "syntactical" reasons, i.e. that do not follow the syntactical conventions above for representing a run of the Minsky machine. The second category (groups 6-7) contains patterns with variables. Typically, these patterns are designed to apply to a wrong computation step, i.e. a pair of consecutive configurations $\#a^{n+1} *^{l} b^{m+1}\#a^{n_1+1} *^{l_1} b^{m_1+1}\#$ where the second one does not encode correctly the result of executing command l with $S_1 = n$, $S_2 = m$. Because of the presence of variables, the difficulty here is that the patterns are potentially applicable to other strings, not necessarily restricted to two consecutive configurations. However, we show that in this case the matched string still cannot be the encoding of a correct execution because of the following two observations:

- If a string contains some configuration $\#a \ldots a * \ldots * b \ldots b\#$ in duplicate, then it cannot represent a finite execution of a program. (Recall that the machine works deterministically and a double occurrence of a configuration in the execution implies that the program does not halt.)
- If a string contains a substring $\#a^{n+1} *^{l} b^{m+1}\#a^{n_1+1} *^{l_1} b^{m_1+1}\#$ where $|n - n_1|$ or $|m - m_1|$ is greater than 1, then the string cannot represent a finite execution of a program. (Recall that a command can change a counter at most by 1).

The rest of the proof is devoted to the construction of the patterns of $\mathcal{R}$. The patterns will be grouped according to the type of "error" in the encoding that they are meant to cover. Each group introduces new patterns and restricts further the set of irreducible instances of w. In the following, x, y, z are variables.

1. Rules for $\models$:

$$x \models y \tag{1}$$

An irreducible string cannot contain $\models$ at an internal position. (Recall that variables are substituted by non-empty words).

$$\models a \tag{2}$$
$$\models b \tag{3}$$
$$\models * \tag{4}$$
$$a \models \tag{5}$$
$$b \models \tag{6}$$
$$* \models \tag{7}$$

$\models$ can be followed and preceded only by $\#$.

2. Rules for syntactical structure:

$$ab \tag{8}$$
$$a\# \tag{9}$$
$$*a \tag{10}$$
$$*\# \tag{11}$$
$$ba \tag{12}$$
$$b* \tag{13}$$

$$\#*\tag{14}$$
$$\#b\tag{15}$$
$$\#\#\tag{16}$$

It should be clear that the instances of w irreducible by rules 1-16 are exactly the strings described by the following regular expression:

$$\models\#(a^+ *^+ b^+\#)^*\models$$

3. Rules for initial configuration. These rules apply to all instances of w that do not start with the encoding of the initial configuration, i.e. do not have the prefix $\models\#a^{2^d+1} * b\#$.

$$\models\#\models\tag{17}$$
$$\models\#*\tag{18}$$
$$\models\#b\tag{19}$$
$$\models\#\#\tag{20}$$
$$\models\#a*\tag{21}$$
$$\ldots$$
$$\models\#a^{2^d-1}*\tag{22}$$
$$\models\#a^{2^d-1}b\tag{23}$$
$$\models\#a^{2^d+1}\tag{24}$$
$$\models\#a^{2^d}b\tag{25}$$
$$\models\#a^{2^d}**\tag{26}$$
$$\models\#a^{2^d}*bb\tag{27}$$

4. Rule for command labels. Suppose that L is the number of commands in $\mathcal{P}$ (which is also the label of the STOP command). The pattern

$$*^{L+1}\tag{28}$$

reduces the strings containing more than L consecutive $*$. These strings are obviously non-correct.

5. Rules for a final configuration. Since the STOP instruction is labeled by L, and S_2 must contain 0 whenever the STOP instruction is to be executed, we have to add the following pattern:

$$*^L\,bb\tag{29}$$

The following pattern applies whenever a final configuration is not followed by $\models$.

$$*^Lb\#a\tag{30}$$

The syntactical form of the instances of w irreducible by rules 1-5 is summarized in the following lemma:

Lemma 2 *Assume that u is a variable, w is $\models u \models$, $d \geq 1$ and let $\sigma(w)$ be an instance of w irreducible by rules 1-5. Then $\sigma(w)$ has the following general form:*

$$\models \# a^{2^d+1} * b \# a^{p_1} *^{q_1} b^{r_1} \# \ldots \# a^{p_i} *^{q_i} b^{r_i} \# \ldots \# a^{p_m} *^{q_m} b^{r_m} \# a^{p_{m+1}} *^{L} b \# \models$$

where $m \geq 0$, $p_i, p_{m+1}, r_i \geq 1$ and $q_i \in \{1, ..., L\}$, for $i \in \{1, ..., m\}$.

6. Rules for instructions of type (i). The rules of this group are introduced for each instruction of type (i) in $\mathcal{P}$.

   Assume that l is an instruction of type (i) that applies to S_1. If a string encodes a correct execution and contains a configuration $\#a^{n+1} *^l b^{m+1} \#$ for some $n, m \geq 0$, then it must be followed by the configuration $\#a^{n+2} *^{l'} b^{m+1} \#$. The patterns below are built to be applicable every time when it is not the case. In other words, a pattern must be applicable to every substring

   $$...\#a^{n+1} *^l b^{m+1} \# a^{n_1+1} *^{l_1} b^{m_1+1} \#...$$

   where either $l_1 \neq l'$ or $n_1 \neq n+1$ or $m_1 \neq m$.

   Here we face the main difficulty caused by the restriction that the introduced patterns should not apply to any piece of the correct execution. The problem is that if a pattern contains a variable, it can match a large part of the subject string and cover more than two consecutive configurations. The problem is solved by using systematically non-linear variables.

   The rules below are schematized using metavariables X, Y, Z where X ranges over $\{\varepsilon, a, aa, aaa, xaax, xaaxa\}$ and Y, Z range over $\{\varepsilon, b, bb, bbb, ybby, ybbyb\}$. Recall that ε denotes the empty string.

   - handling $l_1 \neq l'$

     $$a *^l bY \# Xa *^{l_1} b \tag{31}$$

     Here l_1 ranges over $\{1, \ldots, l'-1\} \cup \{l'+1, \ldots, L\}$. Thus, $6 \times 6 \times (L-1)$ patterns are schematized by 31.

   - handling $n_1 \neq n+1$:
     - case $n_1 \leq n$:

       $$Xa *^l bY \# Xa * \tag{32}$$

     - case $n_1 \geq n+2$:

       $$\#Xa *^l bY \# Xaaa \tag{33}$$

   - handling $m_1 \neq m$:
     - case $m_1 < m$:

       $$a *^l bYZ \# Xa *^{l'} bY \# \tag{34}$$

     - case $m_1 > m$:

       $$a *^l bY \# Xa *^{l'} bYb \tag{35}$$

We summarize the effect of the rules introduced in 6 by the following two lemmas:

Lemma 3 *Assume that u is a variable, w is $\models u\models$ and let $\sigma(w)$ be an instance of w irreducible by rules 1-5. If one of patterns 31-35 applies to $\sigma(w)$, then either $\sigma(w)$ does not encode a correct (finite) execution of the Minsky machine or x matches a word of a^+, y matches a word of b^+, and z matches a word of b^+.*

Proof: We use the fact that in rules 31-35 each variable occurs at a position adjacent to #.

Consider for example rule 31 and let X be substituted by $xaax$ or $xaaxa$. In either case the pattern contains the substring $\#xaax$. Let σ be the matching substitution. By rules 1-5, $\sigma(x)$ starts with a and ends with either a or #. If $\sigma(x)$ contains at least two #, then it must comprise an entire configuration. Since x occurs twice in the pattern, then this configuration must also occur twice in $\sigma(w)$. If $\sigma(x)$ contains exactly one #, then $\sigma(x) = a^p *^q b^r \# a^s$, where $p, q, r \geq 1$, $s \geq 0$. Then the string matched by $\#xaax$ is $\#a^p *^q b^r \# a^{s+2+p} *^q b^r \# a^s$. This string cannot encode an execution step, as counter S_1 is increased by at least 2 which is not possible. Therefore $\sigma(x)$ contains no # and is then composed only of a.

By symmetry, the same reasoning applies to metavariable Y in 31. Moreover, this applies to each variable occurring in a rule, since in every rule each variable has an occurrence adjacent to #.

□

Lemma 4 *Assume that w is $\models u\models$ and let l : `ADD 1 TO` S_j; `GOTO` l' be a command in a program $\mathcal{P}$.*

(a) If one of patterns 31-35 matches a substring of an instance $\sigma(w)$ irreducible by rules 1-5, then $\sigma(w)$ does not encode a correct execution of $\mathcal{P}$.

(b) Conversely, if $\sigma(w)$ contains a substring $\#a^{n+1} *^l b^{m+1} \# a^{n_1+1} *^{l_1} b^{m_1+1} \#$ where either $l_1 \neq l'$ or $n_1 \neq n+1$ or $m_1 \neq m$, then one of the rules from 6 is applicable.

Proof:

(a) By lemma 3, we have to prove the lemma provided that x, y, z are substituted by strings of a^+, b^+, b^+ respectively. Since each of the patterns 31-35 contains the substring $a *^l b$, it can apply only to a substring that correponds to an execution of command l, that is to a substring $\#a^{n+1} *^l b^{m+1} \# a^{n_1+1} *^{l_1} b^{m_1+1} \#$ for some $n, m, n_1, m_1 \geq 0$, $l_1 \in \{0, \ldots, L\}$. It is easy to see that in this case either $l_1 \neq l'$ (if rule 31 applies) or $n_1 \neq n+1$ (if rules 32, 33 apply) or $m_1 \neq m$ (if rules 34, 35 apply). Thus, the matched string cannot be a substring of the encoding of a correct execution.

(b) This is proved by simple case analysis on rules 31-35.

□

If a command l of type (i) applies to counter S_2, the rules are constructed in the same fashion and their correctness can be derived with similar arguments.

7. Rules for instructions of type (ii). The rules of this group are introduced for each instruction of type (ii) in $\mathcal{P}$.

   Assume that l is an instruction of type (ii) that applies to S_1. The strings encoding the correct execution are:

   $$...\#a^{n+1} *^{l} b^{m}\#a^{n} *^{l'} b^{m}\#...$$

   for $n \geq 1$, and

   $$...\#a *^{l} b^{m}\#a *^{l''} b^{m}\#...$$

   otherwise. It is clear that the rules can be built using the same technique as in the previous case. Note that the second string is simpler to treat since the number of 'a' is fixed.

We summarize the construction of the rules above in the following lemma.

Lemma 5 *Assume that w is $\models u \models$, $d \geq 1$ and $\mathcal{R}$ is the set of patterns introduced in 1-7. For a substitution σ, $\sigma(w)$ is reducible w.r.t. $\mathcal{R}$ iff $\sigma(w)$ is not the encoding of a finite execution of $\mathcal{P}$ on $S_1 = 2^d$, $S_2 = 0$.*

Since it is not decidable if the program $\mathcal{P}$ terminates on $S_1 = 2^d$ and $S_2 = 0$, the existence of an irreducible instance of w is not decidable either. This completes the proof of theorem 2. □

It is important to note that most of the technicalities of the proof could be avoided if we allowed a more complex word w. In fact, adding into w another linear variable would make the undecidability proof much simpler. However, we found it interesting to construct a proof for a pattern w the simplest possible. Note that the problem is trivially decidable for $w = au$ (symmetrically, for $w = ua$) where $a \in A$, $u \in \mathcal{X}$, by the following observation: w is ground reducible iff for every letter $b \in A$, the word ab is reducible.

5 Related Works

In this paper we have studied the ground reducibility problem for word rewriting systems with variables and, in particular, proved this problem undecidable in the general case. From a logical point of view, the ground reducibility problem is expressed by a formula in the positive first-order $\forall\exists$-theory of free semigroups. Indeed, assume that we have an alphabet A, and let $\bar{x}$ be the variables of $w \in (A \cup \mathcal{X})^+$ and $\bar{y}$ be the variables occurring in a rewriting system $\mathcal{R} = \{v_1, \ldots, v_n\}$. The ground reducibility of w w.r.t. $\mathcal{R}$ is equivalent to the validity of the following formula in the free semigroup generated by A:

$$\forall\bar{x}\exists u, z\exists\bar{y} \bigvee_{i=1}^{n} w[\bar{x}] = u \cdot v_i[\bar{y}] \cdot z$$

The number of universal quantifiers is equal here to the number of variables in w and the number of existential quantifiers is two more than the maximal number of variables in the rules of $\mathcal{R}$. In particular, the formula corresponding to the proof of theorem 2 contains one universal and five existential quantifiers. In [9] it was proved that the positive first-order $\forall\exists$-theory of free semigroups is in general undecidable. Since we consider

a very special form of positive $\forall\exists$-formulae (which, among other restrictions, do not contain conjunction), our result can be regarded as a reinforcement of that of [9] for this particular fragment of the general positive $\forall\exists$-theory of free semigroups. The undecidable theory constructed in [9] has a single universal and four existential quantifiers. An extra existential quantifier that we have in our proof may be viewed as a price for restricting the theory.

A result by Treinen [12] (theorem 8) about the undecidability of the $\exists\forall$-fragment of a ground term algebra modulo associativity has to be mentioned. Treinen considers a term algebra over a signature of one constant and two binary functions of which one is associative, and $\exists\forall$-formulae with negations. However, the author seems to have been unaware of the result by Marchenko mentioned above. The latter is clearly a stronger statement as it does not use any additional non-constant function symbols. We also note that the positive first-order $\exists\forall$-theory of free semigroups is decidable [3].

As it was noted in the introduction, our theorem 2 refines a result from [5] (theorem 8.2) about the undecidability of ground reducibility in the presence of an associative function. The construction used in the proof in [5] is applicable for both associative and associative-commutative case and uses essentially auxiliary non-constant functions. Our encoding is finer as we have managed to do it without any non-constant functions but a single associative binary function (concatenation). However, with the presented technique we have not been able to cope with commutativity.

Acknowledgements: We are grateful to Paliath Narendran for his valuable comments.

References

[1] D. R. Bean, A. Ehrenfeucht, and G.F. McNulty. Avoidable patterns in strings of symbols. *Pacific Journal of Mathematics*, 85(2):261–294, 1979.

[2] R. V. Book. Thue systems as rewriting systems. *Journal of Symbolic Computation*, 3(1 & 2):39–68, 1987.

[3] V.G. Durnev. Positive theory of a free semigroup. *Dokl. Akad. Nauk SSSR*, 11(4):772–774, 1973. in Russian.

[4] D. Hofbauer and M. Huber. Computing linearizations using test sets. In M. Rusinowitch and J.L. Rémy, editors, *Proceedings 3rd International Workshop on Conditional Rewriting Systems, Pont-à-Mousson (France)*, pages 145–149. CRIN and INRIA-Lorraine, 1992.

[5] D. Kapur, P. Narendran, D. Rosenkrantz, and H. Zhang. Sufficient-completeness, quasi-reducibility and their complexity. Technical Report 87-27, Dept. of Computer Science, State University of New York at Albany, 1987.

[6] D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1987.

[7] G. Kucherov and Tajine M. Decidability of regularity and related properties of ground normal form languages. In M. Rusinowitch and J.L. Rémy, editors, *Proceed-*

ings 3rd International Workshop on Conditional Rewriting Systems, Pont-à-Mousson (France), pages 150–156. CRIN and INRIA-Lorraine, 1992.

[8] G. S. Makanin. Algorithmic decidability of the rank of constant free equations in a free semigroup. *Dokl. Akad. Nauk. SSSR 243*, 243, 1978.

[9] S.S. Marchenko. Undecidability of the positive $\forall\exists$-theory of a free semigroup. *Sibirskii Matematicheskii Zhurnal*, 23(1):196–198, 1982. in Russian.

[10] M. L. Minsky. Recursive unsolvability of post's problem of "tag" and other topics in theory of turing machines. *Annals of Mathematics*, 74(3):437–455, November 1961.

[11] D. Plaisted. Semantic confluence tests and completion methods. *Information and Control*, 65:182–215, 1985.

[12] Ralf Treinen. A new method for undecidabiilty of first order theories. Technical Report A 09/90, Universität des Saarlandes, Fachbereich Informatik, 1990.

[13] S. Vágvölgyi and R. Gilleron. For a rewriting system it is decidable whether the set of irreducible ground terms is recognizable. *Bulletin of European Association for Theoretical Computer Science*, 48:197–209, October 1992.

Unité de Recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)

Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)
Unité de Recherche INRIA Sophia Antipolis 2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399