The Complexity of Testing Ground Reducibility for Linear Word Rewriting Systems with Variables

Gregory Kucherov and Michaël Rusinowitch

INRIA-Lorraine and CRIN
615, rue du Jardin Botanique, BP 101
54506 Vandœuvre-lès-Nancy, France
email: {kucherov,rusi}@loria.fr

Abstract. In [9] we proved that for a word rewriting system with variables $\mathcal{R}$ and a word with variables w, it is undecidable if w is ground reducible by $\mathcal{R}$, that is if all the instances of w obtained by substituting its variables by non-empty words are reducible by $\mathcal{R}$. On the other hand, if $\mathcal{R}$ is linear, the question is decidable for arbitrary (linear or non-linear) w. In this paper we futher study the complexity of the above problem and prove that it is *co-NP*-complete if both $\mathcal{R}$ and w are restricted to be linear. The proof is based on the construction of a deterministic finite automaton for the language of words reducible by $\mathcal{R}$. The construction generalizes the well-known Aho-Corasick automaton for string matching against a set of keywords.

1 Introduction

This paper continues with the study undertaken in [9] where the following question has been proved to be generally undecidable: Given a set of words with variables $\mathcal{R}$ and a word with variables (subject pattern) w, do all the instances of w obtained by substituting its variables by non-empty words contain an instance of a word from $\mathcal{R}$ as a factor? From the point of view of rewriting systems [2], the question amounts to testing a well-known *ground reducibility* property [12] which is at the core of the *inductive completion* methods for proving theorems in the initial model of equational specifications. Here $\mathcal{R}$ is identified with the set of left-hand sides of a rewriting system, and the rewriting systems under consideration, called *word rewriting systems with variables* (WRSV), are rewriting systems over a signature of a single associative binary symbol (concatenation) and a finite set of constants (letters). The undecidability result above strengthens a result from [6] of undecidability of ground reducibility problem for term rewriting systems over an associative function and a number of non-associative ones. On the other hand, the result can be regarded as the undecidability result for a particular fragment of positive $\forall\exists$-theory of free semigroups, and in this sense generalizes the result of [11].

If the rewriting system $\mathcal{R}$ is restricted to be left-linear, the ground reducibility problem is decidable regardless of the linearity of w. In this paper we study the complexity of the problem and prove that it is *co-NP*-complete when w is restricted to be linear too. This case can be viewed as the inclusion problem for regular languages represented by regular expressions of a particular class, the problem known to be *PSPACE*-complete for general regular expressions [3]. Proving that the problem is in *co-NP* requires an analysis of the structure of a deterministic finite automaton recognizing the words reducible by $\mathcal{R}$. The key idea is to prove that although the total number of states in such an automaton is exponential on the size of $\mathcal{R}$, the length of a loop-free path going from the initial to a final state can be bounded polynomially. This allows us to construct a non-deterministic polynomial-time algorithm for checking the existence of an irreducible instance of a given subject word w. The automaton construction generalizes the well-known Aho-Corasick automaton for string matching against a set of keywords. Using similar techniques we prove that the related problem of testing finiteness of the set of irreducible words is also *co-NP*-complete.

2 Notations

Given a finite alphabet A and an alphabet of variables $\mathcal{X}$, we consider words over $A \cup \mathcal{X}$ called *words with variables* (or *patterns*) and words over A called simply *words* (or *strings*). A pattern is called *linear* if every variable occurs at most once in it. A substitution σ is a mapping from $\mathcal{X}$ to A^+.[1] It can be extended to a homomorphism from $(\mathcal{X} \cup A)^+$ to A^+ (regarded as monoids) such that $\sigma(a) = a$ for every $a \in A$.

Word concatenation will be denoted either by $\cdot$ or just by jaxtaposition. ε denotes the empty string and $|w|$ the length of a word 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 $\{p_1, \dots, p_2 - 1\}$.

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 comes from the 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 sometimes we call elements of $\mathcal{R}$ *rules* without adding "the left-hand side of". Similarly, $NF(\mathcal{R})$ stands for the set of all *normal forms* (= irreducible words) for $\mathcal{R}$. A word with variables w is called *ground reducible* by $\mathcal{R}$ iff $Inst(\{w\}) \subseteq Red(\mathcal{R})$, or equivalently, iff $Red(\{w\}) \subseteq Red(\mathcal{R})$.

3 Preliminary results

In [9] we proved the following result.

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

Theorem 1 *For a WRSV $\mathcal{R}$ and a word with variables w, it is undecidable if w is ground reducible by $\mathcal{R}$.*

We even showed that the problem remains undecidable for a fixed and very simple word $w = axa$ where a is a letter and x a variable.

If $\mathcal{R}$ consists of linear patterns, every $v \in \mathcal{R}$ can be written as $v = u_0 x_1 u_1 x_2 \dots x_n u_n$ where $u_0, u_n \in A^*$, $u_i \in A^+$, $1 \leq i \leq n-1$, $x_i \in \mathcal{X}$, $1 \leq i \leq n$ and all x_i are different. Obviously in this case $Red(\{v\}) = A^* u_0 A^+ u_1 A^+ \dots A^+ u_n A^*$ is a regular language and so is $Red(\mathcal{R}) = \bigcup_{v \in \mathcal{R}} Red(\{v\})$. If w is a linear pattern in addition, testing ground reducibility of w by $\mathcal{R}$ amounts to testing the inclusion of regular languages which is of course decidable.

We show now that the problem remains decidable for linear $\mathcal{R}$ and arbitrary (non-linear) w. We need a notation to identify positions in $\sigma(w)$. Assume that $w = u_0 x_1 u_1 x_2 \dots x_n u_n$ (where variables x_i may be equal). If p_i is the position of x_i then by p_i^σ we denote the position in $\sigma(w)$ which corresponds to the beginning of the substring corresponding to x_i. Formally, we define recurrently $p_1^\sigma = p_1$, and $p_{i+1}^\sigma = p_i^\sigma + |\sigma(x_i)| + |u_i|$ for $2 \leq i \leq n$.

Lemma 1 *For 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 [7, 10]. 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. 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 $\sigma(w)$.

Let $\mathcal{A}$ be a deterministic automaton recognizing $NF(\mathcal{R})$. To each position p in a word $u \in NF(\mathcal{R})$ the automaton associates a state denoted $\mathcal{A}(u, p)$. We will use the usual pumping lemma trick: if p_1, p_2 are positions in u, $p_1 < p_2$ and $\mathcal{A}(u, p_1) = \mathcal{A}(u, p_2)$, then $u[p_1 \leftarrow p_2]$ is also recognized by $\mathcal{A}$ and therefore is not $\mathcal{R}$-reducible.

We show now that $C(\mathcal{R}, w)$ can be set to $|\mathcal{A}|^n$, where $|\mathcal{A}|$ is the number of states of $\mathcal{A}$ and n is the maximal number of occurrences 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 $p_1, \dots, p_m$ in w ($m \leq n$). The idea is to find two distinct positions p', p'' in $\sigma(x)$ 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 choose p', p'' that satisfy the following property: for every j, $1 \leq j \leq m$, $\mathcal{A}(\sigma(w), p_j^\sigma + p') = \mathcal{A}(\sigma(w), p_j^\sigma + p'')$. Note that with every position p in $\sigma(x)$ we can associate a m-tuple of states $< \mathcal{A}(\sigma(w), p_1^\sigma + p), \dots, \mathcal{A}(\sigma(w), p_m^\sigma + p) >$. It is clear that there are at most $|\mathcal{A}|^m$ different tuples of this form. Since p has at least $|\mathcal{A}|^n + 1$ possible values where $n \geq m$, by the pigeon hole principle we conclude that positions p', p'' with the desired property must exist. $\square$

It follows from the proof above that if w is linear, $C(\mathcal{R}, w)$ is just the number of states of $\mathcal{A}$. However, it is easy to see that in this case the proof remains valid if $C(\mathcal{R}, w)$ is taken to be a constant which bounds *the number of states along any loop-free path in $\mathcal{A}$ going from the initial to a final state*. This refinement, important for the rest of the paper, is summarized as follows.

Corollary 1 *Let $\mathcal{R}$ be a linear WRSV and w a linear pattern. Assume that $\mathcal{A}$ is a deterministic automaton recognizing $NF(\mathcal{R})$ and K is the maximal length of a loop-free path going from the initial to a final state in $\mathcal{A}$. If w is not ground reducible by $\mathcal{R}$, then there is an irreducible instance $\sigma(w)$ such that $|\sigma(x)| \leq K$ for every variable x of w.*

4 Complexity Results

4.1 Complexity of Testing Ground Reducibility

Now we give a complexity analysis of the ground reducibility problem for a linear WRSV $\mathcal{R}$ and a linear subject pattern w. We show, namely, that this problem is *co-NP*-complete. As usual, the proof consists of two parts. We first prove that the problem is *co-NP*-hard by reducing to it the MONOTONE-ONE-IN-THREE-SAT problem.

Lemma 2 *Testing ground reducibility of a linear subject pattern by a linear WRSV is co-NP-hard.*

Proof: Let X be a finite set of variable symbols and $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m$ be a conjunction of clauses each consisting of 3 variables of X (positive literals). It is known [3] that the following problem, labeled MONOTONE-ONE-IN-THREE-SAT, is *NP*-complete: Given $\mathcal{C}$, does there exist a truth assignment $\sigma : X \to \{\mathtt{t}, \mathtt{f}\}$ such that every clause $\mathcal{C}_i$ contains exactly one variable mapped to $\mathtt{t}$ under σ?

Assume that $\{x_1, \ldots, x_n\}$ is the set of variables in $\mathcal{C}$. We encode $\mathcal{C}$ into the following string over the alphabet $A = \{1, \ldots, n\} \cup \{\mathtt{t}, \mathtt{f}\} \cup \{\#\}$ and a variable set $\mathcal{Y} = \{y_1, y_2, \ldots\}$:

$$w = \#\mathbf{C}_1\#\mathbf{C}_2\#\ldots\#\mathbf{C}_n\#$$

where each $\mathbf{C}_i$ is obtained from $\mathcal{C}_i$ by replacing an occurrence of a variable x_i in $\mathcal{C}_i$ by two symbols $y_j\, i$ where y_j is a fresh variable occurring nowhere else in w. (Two different occurrences of x_i in $\mathcal{C}$ are replaced by $y_j\, i$ and $y_k\, i$ where $y_j \neq y_k$.) By construction, w is a linear pattern.

We construct now a linear WRSV $\mathcal{R}$ over A and $\mathcal{X} = \{x, \ldots\}$ which applies to all instances of w but those which correspond to the solutions of MONOTONE-ONE-IN-THREE-SAT for $\mathbf{C}$.

Firstly, we put into $\mathcal{R}$ the following two patterns:

$$\#\# \tag{1}$$

$$\#x_1\#\ldots\#x_{n+1}\# \tag{2}$$

These patterns reduce any instance of w that contains more than $n+1$ occurrences of # and thus guarantee that every variable in an irreducible instance must be substituted by a string from $(\{\mathtt{t},\mathtt{f}\} \cup \{1,\ldots,n\})^+$.

Then, for every $i, j,\ 1 \le i, j \le n$, we add the pattern

$$ij \tag{3}$$

Also, we add the four patterns

$$\mathtt{tt},\ \mathtt{tf},\ \mathtt{ft},\ \mathtt{ff} \tag{4}$$

To "prevent" variables in w to take values other than $\mathtt{t}, \mathtt{f}$ we add for every $i_1, i_2, i_3, i_4,\ 1 \le i_1, i_2, i_3, i_4 \le n$, the eight patterns schematized by the expression

$$i_1\{\mathtt{t},\mathtt{f}\}i_2\{\mathtt{t},\mathtt{f}\}i_3\{\mathtt{t},\mathtt{f}\}i_4 \tag{5}$$

Patterns (3)-(5) "force" every variable y_j in w to take one of the values $\{\mathtt{t},\mathtt{f}\}$ under σ.

For every $i,\ 1 \le i \le n$, we add the following two patterns. They make reducible any instance of w in which two variables corresponding to the same variable x_i in $\mathcal{C}$ are substituted by different values $\{\mathtt{t},\mathtt{f}\}$.

$$\mathtt{t}ix\mathtt{f}i \tag{6}$$
$$\mathtt{f}ix\mathtt{t}i \tag{7}$$
$$\mathtt{t}i\mathtt{f}i \tag{8}$$
$$\mathtt{f}i\mathtt{t}i \tag{9}$$

Finally, for every $i_1, i_2, i_3\ 1 \le i_1, i_2, i_3 \le n$, we add the patterns

$$\mathtt{f}i_1\mathtt{f}i_2\mathtt{f}i_3 \tag{10}$$
$$\mathtt{t}i_1\mathtt{t}i_2\mathtt{f}i_3 \tag{11}$$
$$\mathtt{t}i_1\mathtt{f}i_2\mathtt{t}i_3 \tag{12}$$
$$\mathtt{f}i_1\mathtt{t}i_2\mathtt{t}i_3 \tag{13}$$
$$\mathtt{t}i_1\mathtt{t}i_2\mathtt{t}i_3 \tag{14}$$

Clearly, the instances of w that remain irreducible by the patterns above correspond exactly to the solutions of MONOTONE-ONE-IN-THREE-SAT for **C**. We have constructed $\mathcal{O}(n^4)$ patterns of constant length and two patterns (including w) of length $\mathcal{O}(n)$ over an alphabet of $\mathcal{O}(n)$ symbols. It is obvious that the whole construction can be done in polynomial time. □

We are now to prove that the existence of an irreducible instance of a linear pattern w w.r.t. a linear WRSV $\mathcal{R}$ can be tested in polynomial time on the non-deterministic Turing machine. In order to simplify the presentation of this part we will allow variables to be substituted by an empty string. We leave the reader to make sure that this assumption does not affect the complexity.

Throughout the rest of this section we assume that $\mathcal{R} = \{p_1, \dots, p_n\}$ is a set of patterns where each p_j is of the form $u_1^j x_1^j u_2^j \dots x_{n_j-1}^j u_{n_j}^j$ where $u_i^j \in A^+$, $1 \leq i \leq n_j$ and $x_i^j \in \mathcal{X}$, $1 \leq i \leq n_j - 1$ are pairwise distinct. (Note that a linear variable at the beginning and/or at the end of a pattern can be omitted.) A non-deterministic algorithm that tests whether there exists an irreducible instance of w w.r.t. $\mathcal{R}$ consists of the following two steps.

1. guess a substitution σ assigning to every variable a string no longer than K, where K is the constant from corollary 1,
2. test if $\sigma(w)$ is irreducible by $\{p_1, \dots, p_n\}$

Let $C = \max\{|u_i^j| \, |1 \leq j \leq n, 1 \leq i \leq n_j\}$ and $N = \max\{n_j \, |1 \leq j \leq n\}$. Step 2 can be done deterministically in time $\mathcal{O}(n(|\sigma(w)| + CN))$ by using for each p_j the Knuth-Morris-Pratt string matching algorithm successively for $u_1^j, \dots, u_{n_j}^j$. In order to prove that the whole algorithm is in polynomial time it is sufficient to show that K can be bounded polynomially on the size of $\mathcal{R}$. In this case the size of a guess at step 1 and the run time of step 2 would be polynomial on $(|\mathcal{R}| + |w|)$.

Recall that K is the length of the longest loop-free path going from the initial to a final state in some deterministic automaton recognizing $NF(\mathcal{R})$. Note that we can equivalently reason about a deterministic automaton for $Red(\mathcal{R})$ since it can be obtained from the one for $NF(\mathcal{R})$ by changing the set of final states to its complement. From semantical considerations it is clear that in a deterministic automaton for $Red(\mathcal{R})$:

– every transition from a final state leads to a final state,
– from every reachable state there is a path to a final state.

Therefore, K can be taken to be the length of the longest loop-free path in some deterministic automaton recognizing $Red(\mathcal{R})$.

In the rest of this section we construct a deterministic automaton for $Red(\mathcal{R})$ for which we show that although the total number of states is exponential on $|\mathcal{R}|$, the number of states along a loop-free path can be bounded polynomially.

Example 1 For $k > 0$, consider the system

$$\mathcal{R} = \{\#a\#x\#a\#,\ \#aa\#x\#aa\#, \dots, \#a^k\#x\#a^k\#\}$$

over the two-letter alphabet $A = \{a, \#\}$. It can be shown that the minimal deterministic automaton for $Red(\mathcal{R})$ has the number of states exponential on k. Informally, if the automaton reaches some non-final state after reading a word w, then this state should "memorize" the set $\{i \,|\, 1 \leq i \leq k, \#a^i\# \text{ is a subword of } w\}$. States corresponding to different sets cannot be factorized since for any two of them, there is a word which leads to a final state from one but not from another. The number of different such sets is 2^k.

On the other hand, the longest loop-free paths are of polynomial length. It can be proved that the words spelled out by the longest loop-free paths in the automaton are of the form $w = \#a^{i_1}\#\#a^{i_2}\# \dots \#a^{i_k}\#\#a^k\#$, where $(i_1, i_2, \dots, i_k)$

is an arbitrary permutation of $(1, \ldots, k)$. The proof consists of two parts. First we show that the automaton does not go twice through the same state during its run on w. To show this, we prove that for any two distinct prefixes w_1, w_2 of w, there is a word $v \in A^*$ such that $w_1 v \in Red(\mathcal{R})$ and $w_2 v \notin Red(\mathcal{R})$, or vice versa. This implies that no two prefixes of w take the automaton to the same state. At the second step, we show that any word longer than w has two distinct prefixes w_1, w_2 such that for every $v \in A^*$, $w_1 v \in Red(\mathcal{R})$ iff $w_2 v \in Red(\mathcal{R})$. This means that any such word makes the automaton visit twice the same state. Both steps of the proof can done by exhaustive case analysis. We omit further details.

In conclusion, the length of the longest loop-free paths is $\frac{k^2+5k+4}{2}$. □

To explain the structure of the automaton let us start with a very particular case when every pattern $p_i \in \mathcal{R}$ is just a string $v_i \in A^+$. We come up then with a well-known problem of matching against a set of keywords [1]. The well-known Aho-Corasick algorithm is a generalization of the Knuth-Morris-Pratt algorithm to the multiple-keyword case [1]. Similar to the Knuth-Morris-Pratt algorithm, the Aho-Corasick algorithm preprocesses (in time $\mathcal{O}(|\mathcal{R}|)$) the set $\mathcal{R}$ into an automaton which allows one to perform pattern matching by scanning the input string in linear time without backtracking. Let us recall very briefly the idea of the construction. Think of the algorithm as scanning the input string and moving a pointer in each v_i. Clearly, a state in the automaton is associated with a combination of pointer positions. However, it is not necessary to consider (a potentially exponential number of) all possible combinations, as the following argument shows. The position of each pointer is uniquely determined by the suffix w of length $\max\{|v_i| \; |1 \leq i \leq n\}$ of the scanned part of the input string. The pointer position in v_i is then defined as the longest prefix of v_i which is a suffix of w. Moreover, it is sufficient to know the longest suffix of w which is at the same time a prefix of some v_i. But this shows that a state can be identified with a prefix of some v_i which shows that the number of states is bounded by $|\mathcal{R}|$. We essentially use this idea in the construction below.

We describe a deterministic automaton $\mathcal{A}$ that recognizes the strings matched by at least one of the patterns. Let us first introduce some notations. Given a word w, *pref*(w) (respectively *suff*(w)) denotes the set of prefixes (respectively suffixes) of w. ε denotes the empty string. Given two words v, w, $S(v, w)$ stands for the longest word from *suff*$(v) \cap$ *pref*(w). Finally, if q is a state and v a word, $q \cdot v$ denotes the state reached by $\mathcal{A}$ from the state q after processing the word v.

The set of states Q of $\mathcal{A}$ is a set of triples

$$< (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$$

where $1 \leq i_j \leq n_j + 1$, $\pi \in \bigcup_{j=1}^{n} pref(u^j_{i_j})$ and $\mu_j \in pref(u^j_{i_j}) \cup \{*\}$.
The initial state of the automaton is :

$$q_0 =< (1, \ldots, 1), \varepsilon, (\varepsilon, \ldots, \varepsilon) >$$

Assume that $a \in A$, $q =< (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$, and $i_j \leq n_j$ for all j, $1 \leq j \leq n$. We define $q \cdot a =< (i'_1, \ldots, i'_n), \pi', (\mu'_1, \ldots, \mu'_n) >$, where $i'_1, \ldots, i'_n, \pi', \mu'_1, \ldots, \mu'_n$ are computed as follows. For every j, $1 \leq j \leq n$, first compute

$$\alpha_j = \begin{cases} S(\pi a, u^j_{i_j}) & \text{if } \mu_j = * \\ S(\mu_j a, u^j_{i_j}) & \text{if } \mu_j \neq * \end{cases} \tag{15}$$

There are two cases:

1. (*local transition*) if there is no j, $1 \leq j \leq n$ such that $\alpha_j = u^j_{i_j}$, then
   (a) $i'_j = i_j$ for all j, $1 \leq j \leq n$,
   (b) π' is the longest string of $\{\alpha_1, \ldots, \alpha_n\}$,
   (c) for every j, $1 \leq j \leq n$,

$$\mu'_j = \begin{cases} * & \text{if } \alpha_j = S(\pi', u^j_{i_j}), \\ \alpha_j & \text{otherwise} \end{cases} \tag{16}$$

2. (*global transition*) if there exists j, $1 \leq j \leq n$ such that $\alpha_j = u^j_{i_j}$, then
   (a) for every j, $1 \leq j \leq n$, $i'_j = \begin{cases} i_j + 1 & \text{if } \alpha_j = u^j_{i_j}, \\ i_j & \text{otherwise} \end{cases}$
   (b) π' is the longest string of $\{\alpha_j \mid 1 \leq j \leq n,\ \alpha_j \neq u^j_{i_j}\}$,
   (c) for every j, $1 \leq j \leq n$,

$$\mu'_j = \begin{cases} \varepsilon & \text{if } \alpha_j = u^j_{i_j}, \\ * & \text{if } \alpha_j \neq u^j_{i_j} \text{ and } \alpha_j = S(\pi', u^j_{i_j}), \\ \alpha_j & \text{otherwise} \end{cases} \tag{17}$$

If a (global) transition results in a state $q =< (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$ such that $i_j = n_j + 1$ for some j, $1 \leq j \leq n$, then q is a final state. Every transition from a final state leads to the same state. Clearly, the constructed automaton is deterministic and complete.

Let us explain informally the construction above. As for the Aho-Corasick algorithm, think of the automaton as simulating the process of moving a pointer in every pattern $p_j = u^j_1 x^j_1 u^j_2 \ldots x^j_{n_j-1} u^j_{n_j}$. At each moment the pointer is located in some $u^j_{i_j}$, and the pointed prefix of $u^j_{i_j}$, say ν_j, is a suffix of the scanned part of the input string. After reading a letter a from the input, the pointer moves one position right if a is the letter which follows ν_j in $u^j_{i_j}$, and moves left or stays at the same position otherwise. In the latter case the pointed prefix becomes equal to the longest prefix of $u^j_{i_j}$ which is a suffix of $\nu_j a$. In both cases, the new pointed prefix is $S(\nu_j a, u^j_{i_j})$. Once the pointer gets to the end of $u^j_{i_j}$, the following word $u^j_{i_j+1}$ is entered, that is the pointer is placed at the beginning of it.

The first component $(i_1, \ldots, i_n)$ of the state of the automaton indicates, for every p_j, the word $u^j_{i_j}$ that the pointer is currently located in. We call i_j *the* j*-th coordinate* of the state. The second component π is maintained to be the longest among all pointed prefixes. Unlike the Aho-Corasick algorithm, π does not generally determine the pointer position in each of $u^1_{i_1}, \ldots, u^n_{i_n}$ because they have not been generally entered at the same moment. Formally, saying that π determines the pointer position in $u^j_{i_j}$ means that the pointed prefix is equal to $S(\pi, u^j_{i_j})$. In order to keep track of the pointer positions, a third component $(\mu_1, \ldots, \mu_n)$ is added to the state. If π determines the pointer position in $u^j_{i_j}$, then the corresponding μ_j is set to $*$, otherwise μ_j is assigned the pointed prefix of $u^j_{i_j}$. The states of the automaton are defined recursively. To compute a "new" state $q \cdot a$ from a "current" state q, auxiliary words $\alpha_1, \ldots, \alpha_n$ are first computed which correspond exactly to the new pointer positions in $u^1_{i_1}, \ldots, u^n_{i_n}$.

The next two lemmas show the correctness of the construction, i.e. that the automaton recognizes precisely the words reducible by $\mathcal{R}$.

Lemma 3 *Assume that $v \in A^*$ and $q = q_0 \cdot v$. Assume that no proper prefix of v is accepted by the automaton (i.e. either q is not final or it is a final state reached by the automaton for the first time during its run on v).*

(1) If $q = < (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$, then for every j, $1 \le j \le n$, there exists a decomposition

$$v = \beta^j_1 u^j_1 \beta^j_2 \ldots u^j_{i_j - 1} \beta^j_{i_j}, \quad \beta^j_1, \ldots, \beta^j_{i_j} \in A^* \tag{18}$$

where $\beta^j_1, \beta^j_2, \ldots, \beta^j_{i_j}$ satisfy the following properties

(i) for every k, $1 \le k \le i_j - 1$, u^j_k does not occur in $\beta^j_k u^j_k$ as a factor except at the suffix position,

(ii) if $i_j \neq n_j + 1$, then $u^j_{i_j}$ does not occur in $\beta^j_{i_j}$ as a factor,

(iii) if $i_j \neq n_j + 1$, then $S(\beta^j_{i_j}, u^j_{i_j}) = \begin{cases} S(\pi, u^j_{i_j}) & \text{if } \mu_j = * \\ \mu_j & \text{if } \mu_j \neq * \end{cases}$

(2) Conversely, for every j, $1 \le j \le n$, let i_j, $1 \le i_j \le n_j$, and $\beta^j_1, \ldots, \beta^j_{i_j}$ be such that v admits decomposition (18) that satisfies conditions (i),(ii). Then $q = < (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$, where $\pi, \mu_1, \ldots, \mu_n$ verify the following conditions

(iv) π is the longest string of $\{S(\beta^j_{i_j}, u^j_{i_j}) \mid 1 \le j \le n\}$,

(v) $\mu_j = \begin{cases} \varepsilon & \text{if } \beta^j_{i_j} = \varepsilon \\ * & \text{if } \beta^j_{i_j} \neq \varepsilon \text{ and } S(\pi, u^j_{i_j}) = S(\beta^j_{i_j}, u^j_{i_j}) \\ S(\beta^j_{i_j}, u^j_{i_j}) & \text{otherwise} \end{cases}$

Proof: First we note that part (2) of the lemma is stated correctly since a decomposition of v satisfying (i),(ii) is unique and therefore i_j's and $\beta^j_{i_j}$'s are well-defined.

We use induction on the length of v.

(1) For $v = \varepsilon$ the lemma trivially holds. Assume that the lemma holds for a word v and $q_0 \cdot v = < (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$. Let $a \in A$ and

$q_0 \cdot va =< (i'_1, \ldots, i'_n), \pi', (\mu'_1, \ldots, \mu'_n) >$. We have to show that va can be decomposed according to the lemma where $i_1, \ldots, i_n, \pi, \mu_1, \ldots, \mu_n$ are replaced by $i'_1, \ldots, i'_n, \pi', \mu'_1, \ldots, \mu'_n$ respectively.

Consider the a-transition from $q_0 \cdot v$ to $q_0 \cdot va$ and suppose it is a local transition, i.e. $i'_j = i_j$ for all j. By assumption, $q_0 \cdot v$ is not final and thus $q_0 \cdot va$ is not final either (i.e. $i'_j \neq n_j$ for all j). Take some j, $1 \leq j \leq n$. By induction hypothesis $v = \beta^j_1 u^j_1 \beta^j_2 \ldots u^j_{i_j-1} \beta^j_{i_j}$, and conditions (i)-(iii) are verified. Let us show that the decomposition $va = \beta^j_1 u^j_1 \beta^j_2 \ldots u^j_{i_j-1} \delta^j_{i_j}$, where $\delta^j_{i_j} = \beta^j_{i_j} a$ satisfies the lemma. Condition (i) is trivially verified. By induction hypothesis (condition (iii))

$$S(\beta^j_{i_j}, u^j_{i_j}) = \begin{cases} S(\pi, u^j_{i_j}) & \text{if } \mu_j = * \\ \mu_j & \text{if } \mu_j \neq * \end{cases}$$

This implies that

$$S(\beta^j_{i_j} a, u^j_{i_j}) = \begin{cases} S(\pi a, u^j_{i_j}) & \text{if } \mu_j = * \\ S(\mu_j a, u^j_{i_j}) & \text{if } \mu_j \neq * \end{cases}$$

The expression on the right is exactly α_j defined by (15). Since the transition is local, then α_j is a proper prefix of $u^j_{i_j}$, and therefore $u^j_{i_j}$ is not a suffix of $\beta^j_{i_j} a$. Thus, condition (ii) is also verified. By reading (16) from right to left, we have

$$\alpha_j = \begin{cases} S(\pi', u^j_{i_j}) & \text{if } \mu'_j = * \\ \mu'_j & \text{if } \mu'_j \neq * \end{cases}$$

Thus,

$$S(\delta^j_{i_j}, u^j_{i_j}) = \begin{cases} S(\pi', u^j_{i_j}) & \text{if } \mu'_j = * \\ \mu'_j & \text{if } \mu'_j \neq * \end{cases}$$

which proves condition (iii).

Assume now that the transition under consideration is global. For those j's that satisfy $i_j = i'_j$, the same decomposition and proof as in the case of local transition apply. Consider j such that $\alpha_j = u^j_{i_j}$ and $i'_j = i_j + 1$. Then the decomposition $va = \beta^j_1 u^j_1 \beta^j_2 \ldots u^j_{i_j-1} \delta^j_{i_j} u^j_{i_j} \beta^j_{i_j+1}$ satisfies the lemma, where $\delta^j_{i_j} u^j_{i_j} = \beta^j_{i_j} a$ and $\beta^j_{i_j+1} = \varepsilon$. Note that this decomposition is correct since $S(\beta^j_{i_j} a, u^j_{i_j}) = u^j_{i_j}$, that is $u^j_{i_j}$ is indeed a suffix of $\beta^j_{i_j} a$. Condition (i) of the lemma follows from the induction hypothesis (condition (ii)) that $\beta^j_{i_j}$ does not contain $u^j_{i_j}$ as a factor. Condition (ii) is trivial as $\beta^j_{i_j+1} = \varepsilon$. Condition (iii) is also trivial as $\mu'_j = \varepsilon$ by (17).

(2) This part can be proved using similar arguments. □

Let $v \in A^*$, $|v| > 0$. For some j, $1 \leq j \leq n$, consider the decomposition of v according to lemma 3. The remarks below follow from the proof above.

Remark 1 *If q is a current state and the last transition was local, then $S(\beta^j_{i_j}, u^j_{i_j}) = \alpha_j$ where α_j's are computed according to (15) and correspond to the last transition.*

Remark 2 *$\beta^j_{i_j} = \varepsilon$ iff the last transition was global and modified the j-th coordinate of the state from $i_j - 1$ to i_j. Otherwise $|\beta^j_{i_j}|$ is equal to the number of transitions made after that modification.*

Lemma 4 *The language accepted by the automaton described above is $Red(\mathcal{R}) = \bigcup_{j=1}^n A^* u^j_1 A^* \ldots A^* u^j_{n_j} A^*$*

Proof: Let $w \in A^*$ be accepted by the automaton. Take the shortest prefix v of w accepted by the automaton. Assume that $q_0 \cdot v = q$, $q =< (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$, and $i_j = n_j + 1$ for some j, $1 \leq j \leq n$. From part (1) of lemma 3 it follows that v can be decomposed as $v = \beta^j_1 u^j_1 \beta^j_2 \ldots \beta^j_{n_j} u^j_{n_j}$ (by remark 2, $\beta^j_{n_j+1} = \varepsilon$). Therefore v is reducible by $p_j = u^j_1 x^j_1 u^j_2 \ldots x^j_{n_j-1} u^j_{n_j}$ and so is w.

If $w \in Red(\mathcal{R})$, take the shortest reducible prefix v of w, and let $p_j = u^j_1 x^j_1 u^j_2 \ldots x^j_{n_j-1} u^j_{n_j}$ be a pattern which applies to v. Find a decomposition $v = \beta^j_1 u^j_1 \beta^j_2 \ldots \beta^j_{n_j} u^j_{n_j}$ such that for every k, $1 \leq k \leq n_j$, u^j_k does not occur in $\beta^j_k u^j_k$ as a factor except at the suffix position. This decomposition can be obtained by taking iteratively for each k, $1 \leq k \leq n_j$, the leftmost occurrence of u^j_k which follows the occurrence of u^j_{k-1}. By part 2 of lemma 3, v takes the automaton to a final state, and therefore w is also accepted. $\square$

The following lemma shows that after a bounded number of steps every μ_j gets equal to $*$ unless the j-th coordinate of the state is changed.

Lemma 5 *Let $v \in A^*$ and $q_0 \cdot v =< (i_1, \ldots, i_n), \pi, (\mu_1, \ldots, \mu_n) >$. Assume that $v = \beta^j_1 u^j_1 \beta^j_2 \ldots u^j_{i_j-1} \beta^j_{i_j}$ is the decomposition of v according to lemma 3 for some j, $1 \leq j \leq n$. Then $|\beta^j_{i_j}| \geq |u^j_{i_j}|$ implies $\mu_j = *$.*

Proof: Since $|\beta^j_{i_j}| > 0$, the last transition did not change the j-th coordinate of the state (remark 2). Together with (16), (17) this implies that proving $\mu_j = *$ amounts to proving $\alpha_j = S(\pi, u^j_{i_j})$ where α_j corresponds to the last transition on the path induced by v. On the other hand, $\alpha_j = S(\beta^j_{i_j}, u^j_{i_j})$ according to remark 1. Hence, we have to prove that $S(\pi, u^j_{i_j}) = S(\beta^j_{i_j}, u^j_{i_j})$.

Recall that both π and $\beta^j_{i_j}$ is a suffix of v. If π is longer than $\beta^j_{i_j}$, then every suffix of $\beta^j_{i_j}$ is also a suffix of π. On the other hand, since $|\beta^j_{i_j}| \geq |u^j_{i_j}|$, every prefix of $u^j_{i_j}$ which is a suffix of π is also a suffix of $\beta^j_{i_j}$. Therefore, $S(\pi, u^j_{i_j}) = S(\beta^j_{i_j}, u^j_{i_j})$. If $\beta^j_{i_j}$ is longer than π, then every suffix of π is also a suffix of $\beta^j_{i_j}$. On the other hand, π is longer than or equal to $S(\beta^j_{i_j}, u^j_{i_j})$ by definition of π. This implies again $S(\pi, u^j_{i_j}) = S(\beta^j_{i_j}, u^j_{i_j})$. □

Now we are in position to establish a bound for the loop-free paths in the automaton.

Lemma 6 *Assume that* $M = \sum_{j=1}^n n_j$ *and* $C = \max\{|u^j_{i_j}|| 1 \le j \le n, 1 \le i_j \le n_j\}$. *Then the maximal length of a loop-free transition sequence of* $\mathcal{A}$ *is bounded by* $(n+1)MC$.

Proof: Consider an arbitrary loop-free path in the automaton. It is clear that any chain of transitions modifies the first tuple of the state at most $\sum_{j=1}^n (n_j - 1) + 1$ times before reaching an accepting state.

Let us fix the tuple of coordinates to $(i_1, \ldots, i_n)$. By lemma 5 and remark 2, after at most C transitions every μ_j gets equal to $*$ and keeps this value unless i_j is modified. As soon as both the first and the third component is fixed, every state is uniquely associated with the value of π. Since π is a prefix of some word of $u^1_{i_1}, \ldots, u^n_{i_n}$, there are at most nC such states.

To sum up, the length of a loop-free path in the automaton is bounded by $M(C + nC) = (n+1)MC$. □

Thus, the length of a loop-free path in a deterministic automaton which recognizes $Red(\mathcal{R})$ ($NF(\mathcal{R})$) can be bounded polynomially (quadratically) on $|\mathcal{R}|$. In conclusion, we obtain

Lemma 7 *Testing ground reducibility of a linear subject pattern by a linear WRSV is in co-NP.*

Finally, lemmas 2 and 7 prove the main result.

Theorem 2 *Testing ground reducibility of a linear subject pattern by a linear WRSV is co-NP-complete.*

4.2 Complexity of Testing Finiteness of $NF(\mathcal{R})$

We use the technique of the previous section to show that if a WRSV $\mathcal{R}$ is restricted to be linear, the problem of finiteness of the set $NF(\mathcal{R})$ of irreducible words is also *co-NP*-complete.

Assume we are given a linear WRSV $\mathcal{R}$. From lemma 6 the length of all loop-free paths in the automaton $\mathcal{A}$ constructed in the previous section is bounded by a polynomial $p(|\mathcal{R}|)$. This implies that $p(|\mathcal{R}|) - 1$ bounds the length of irreducible words in the case when their number is finite. Conversely, if every word of length $p(|\mathcal{R}|)$ is reducible, then this is trivially the case for all longer words. Thus, to test nondeterministically if $NF(\mathcal{R})$ is infinite, guess a word of the length $p(|\mathcal{R}|)$ and check if it is irreducible. This proves that testing finiteness of $NF(\mathcal{R})$ is in *co-NP*.

Now we prove that the problem is complete for *co-NP*.

Lemma 8 *Testing finiteness of* $NF(\mathcal{R})$ *for a linear WRSV* $\mathcal{R}$ *is co-NP-complete.*

Proof: By the remark above it remains to show that the problem is *co*-NP-hard. We encode a formula $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m$ into a WRSV $\mathcal{R}$ over the alphabet $A = \{1, \ldots, n\} \cup \{\mathtt{t}, \mathtt{f}\} \cup \{\#\}$ and a variable set $\mathcal{X} = \{x, \ldots\}$. For technical reasons we assume that all $\mathcal{C}_i$ are different.

We first modify the WRSV constructed in the proof of lemma 2. We replace pattern 2 by the patterns

$$\mathtt{t}\#, \mathtt{f}\# \tag{19}$$

$$\#i, \quad \text{for all } 1 \leq i \leq n \tag{20}$$

We add further the patterns

$$\#\{\mathtt{t},\mathtt{f}\}i\#, \quad \#\{\mathtt{t},\mathtt{f}\}i\{\mathtt{t},\mathtt{f}\}j\#, \quad \text{for all } 1 \leq i, j \leq n \tag{21}$$

It should be clear that words that remain irreducible are factors of words from the regular language $((\{\mathtt{t},\mathtt{f}\}\{1, \ldots, n\})^3\#)^*$.

Assume that $\mathbf{C}_i$ encodes $\mathcal{C}_i$ in the same way as in the proof of lemma 2. Define $C_i \subseteq (\{\mathtt{t},\mathtt{f}\}\{1, \ldots, n\})^3$ to be the set of all instances of $\mathcal{C}_i$ which can be obtained by applying some truth assignement. We add to $\mathcal{R}$ all the words from

$$(\{\mathtt{t},\mathtt{f}\}\{1, \ldots, n\})^3 \setminus \bigcup_{i=1}^{n} C_i \tag{22}$$

Now, every 6-letter factor of an irreducible word occurring between two $\#$'s belongs to some C_i. Finally, let $\bar{C}_i = (\{\mathtt{t},\mathtt{f}\}\{1, \ldots, n\})^3 \setminus C_i$. We add the patterns

$$v\#v', \quad \text{for all } 1 \leq i \leq n-1, \ v \in C_i, \ v' \in \bar{C}_{i+1} \tag{23}$$

$$v\#v', \quad \text{for all } v \in C_n, \ v' \in \bar{C}_1 \tag{24}$$

If the set of irreducible words w.r.t. the constructed WRSV is infinite, then every sufficiently long irreducible word contains a factor $\#v_1\#v_2\# \ldots \#v_n\#$ that encodes a solution of the MONOTONE-ONE-IN-THREE-SAT problem for $\mathcal{C}$. Conversely, if $\#v_1\#v_2\# \ldots \#v_n\#$ encodes a solution of $\mathcal{C}$, then all the words $(v_1\#v_2\# \ldots \#v_n\#)^*$ are irreducible. We conclude that $NF(\mathcal{R})$ is infinite if and only if $\mathcal{C}$ has a solution, and therefore testing finiteness of $NF(\mathcal{R})$ for a given $\mathcal{R}$ is *co*-NP-hard. □

5 Remarks and Related Works

Note that theorem 2 remains valid even if the subject pattern w is assumed to be fixed. Necessary modifications of the proof of lemma 2 are suggested by the proof of lemma 8.

The ground reducibility problem we have considered in section 4.1 is the inclusion problem for regular languages represented by regular expressions of a particular class. The inclusion problem for general regular languages represented by regular expressions is $PSPACE$-complete [3]. Various complexity results for formal language theory can be found in [5, 4]. For example, it is proven that the inclusion of regular languages $L_1 \subseteq L_2$ remains $PSPACE$-complete even if L_1 is fixed. On the other hand, we are unaware about results on complexity of language inclusion (equivalence) for subclasses of regular languages similar to the one considered in this paper.

Recently we proposed an efficient algorithm for testing the reducibility of a word with respect to a linear WRSV [8]. This problem is equivalent to a string matching problem for a specific set of patterns (strings with *variable length don't-care symbols*), and has various practical applications.

Acknowledgements: We are grateful to Paliath Narendran for giving an initial impulse to this study and to Valentin Antimirov for enlightening discussions about Example 1.

References

1. A. V. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers B. V. (North-Holland), 1990.
2. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers B. V. (North-Holland), 1990.
3. M. Garey and D. Johnson. *Computers and Intractability. A guide to the theory of NP-completeness*. W. Freeman and Compagny, New York, 1979.
4. Harry B. Hunt III and Daniel J. Rosenkrantz. Computational parallels between the regular and context-free languages. *Theoretical Computer Science*, 7(1):99–114, February 1978.
5. Harry B. Hunt III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences*, 12:222–268, 1976.
6. D. Kapur, P. Narendran, D. Rosenkrantz, and H. Zhang. Sufficient-completeness, ground-reducibility and their complexity. *Acta Informatica*, 28:311–350, 1991.
7. D. Kapur, P. Narendran, and H. Zhang. On sufficient completeness and related properties of term rewriting systems. *Acta Informatica*, 24:395–415, 1987.
8. G. Kucherov and M. Rusinowitch. Matching a set of strings with variable length don't cares. In E. Ukkonen, editor, *Proceedings of the 6th Symposium on Combinatorial Pattern Matching*, Helsinki, July 1995. to appear in Lect. Notes Comput. Sci. Series.
9. G. Kucherov and M. Rusinowitch. Undecidability of ground reducibility for word rewriting systems with variables. *Information Processing Letters*, 53:209–215, 1995.
10. G. Kucherov and M. Tajine. Decidability of regularity and related properties of ground normal form languages. *Information and Computation*, 117, 1995. to appear.
11. S.S. Marchenko. Undecidability of the positive $\forall\exists$-theory of a free semigroup. *Sibirskii Matematicheskii Zhurnal*, 23(1):196–198, 1982. in Russian.
12. D. Plaisted. Semantic confluence and completion method. *Information and Control*, 65:182–215, 1985.