Satisfiability of Word Equations with Constants Is in PSPACE

WOJCIECH PLANDOWSKI

Warsaw University, Warsaw, Poland

Abstract. We prove that satisfiability problem for word equations is in PSPACE.

Categories and Subject Descriptors: F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*decision problems*

General Terms: Theory

Additional Key Words and Phrases: String unification, word equations

1. *Introduction*

The satisfiability problem for word equations has a simple formulation: Find out whether or not an input word equation has a solution. The decidability of the problem was proved by Makanin [1977]. His decision procedure has one of the most complicated termination proofs existing in the literature. There were several attempts to simplify it [Abdulrab and Pécuchet 1989; Pécuchet 1981]. The algorithm has been implemented [Abdulrab 1987]. During last 20 years its complexity has been improved several times: 4-NEXPTIME (composition of four exponential functions) [Jaffar 1990; Schulz 1992], 3-NEXPTIME [Kościelski and Pacholski 1996], 2-EXPSPACE (V. Diekert, 1998, Personal communication), EXPSPACE [Gutierrez 1998]. The exact complexity of the algorithm is still not known. The current version of the algorithm, the full version of which can be found in Diekert [2002], is still very complicated.

Recently, another algorithm has been proposed in Plandowski and Rytter [1998]. It works nondeterministically in time polynomial in $n \log N$ where n is the size of an input equation and N is the size of a minimal solution of the input equation.

This article was originally published in *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS'99)*, IEEE Computer Society Press, Los Alamitos, Calif., 1999, pp. 495–500.

This research was supported by KBN grant 4T11C04425.

Author's address: Institute of Informatics, Warsaw University, Banacha 2, 02-097, Warsaw, Poland, e-mail: wojtekpl@mimuw.edu.pl.

It is much more simple than Makanin's algorithm but its full version requires an estimation for N and the algorithm in Plandowski [1994, 1996]. Until 1999, the only estimation for N was a consequence of Makanin's algorithm. The one which can be concluded from the last version of the algorithm is triple exponential [Gutierrez 1998, Corollary 1]. Very recently, a double exponential estimation for N has been proved [Plandowski 1999]. The proof does not use Makanin's approach but uses the optimal bound for the index of periodicity of a minimal solution [Kościelski and Pacholski 1996]. With this estimation, the algorithm in Plandowski and Rytter [1998] places the problem in NEXPTIME.

We propose the third algorithm. The full version of the algorithm requires only a proof of the upper bound for index of periodicity of a minimal solution [Kościelski and Pacholski 1996] and a lemma in Plandowski and Rytter [1998]. Our algorithm is the first one that is proven to work in polynomial space.

The problem is NP-hard [Angluin 1979; Kościelski and Pacholski 1996]. The best algorithms for NP-hard problems run in single exponential deterministic time. Each problem in PSPACE can be solved in single exponential deterministic time. So our algorithm is time-optimal unless faster algorithms are developed for NP-hard problems.

The satisfiability of word equations is a part of Unification Theory [Baader and Snyder 2001]. Quite recently a connection between word equations and string graphs has been found [Schaefer et al. 2002].

2. *The Algorithm—Overview*

Given a word equation e we construct a relation $\rightarrow$ on polynomial size descriptions of possibly singly exponential length word equations. The relation satisfies

$$(a = a) \rightarrow^* e \text{ if and only if } e \text{ has a solution (Lemma 7.4),}$$

where $\rightarrow^*$ is a transitive closure of $\rightarrow$. Here, the equation $a = a$ is a trival one—it does not contain variables. We have $e_1 \rightarrow e_2$ where e_1 and e_2 are polynomial size descriptions of word equations, if e_2 can be obtained from e_1 by a nondeterministic polynomial space procedure described in Section 4.

Our algorithm is nondeterministic:

eq:= $(a = a)$.
while $(eq \neq e)$ **do**
    nondeterministically generate next description eq'
        such that $eq \rightarrow eq'$;
    eq:=eq'

The algorithm is in NPSPACE since it works on short descriptions of equations and because the generation of a next description can be done in NPSPACE. Since NPSPACE=PSPACE the result follows.

The crucial lemma in our considerations is Lemma 7.4 which looks as follows.

LEMMA 7.4. *$(a = a) \rightarrow^* e$ if and only if e has a solution.*

The proof of Lemma 7.4 consists of proving two implications. The one from left to right is a simple one. Its proof is in Section 4. The one from right to left is a difficult one. It is based on generalized factorizations called here *index factorizations of a minimal solution of e*. We construct a sequence of index factorizations of a minimal

solution. Each of the factorizations corresponds to an equation on index-factors called factor equations. In this way, from a sequence of factorizations, we obtain a sequence of factor equations. Each of these equations is equivalent to some ordinary equation that has a polynomial-size description (Lemma 7.1). In this way, we obtain a sequence of polynomial-size descriptions $\{e_l\}_{l\geq 0}$ of ordinary equations. In this sequence, the first description e_0 describes the trivial equation $a = a$, the last one describes e and $e_l \to e_{l+1}$, for $l \geq 0$.

3. *Preliminaries*

The length of a word w is denoted by $|w|$. A word u is a *prefix* of a word w if there is a word v such that $w = uv$. Then we write $v = u^{-1}w$. A word u is a *subword* of a word w if there are words p and r such that $w = pur$. A subword of w starting at position i and ending at position j is denoted by $w[i, j]$. A *period* of a word is a number p such that $w[i] = w[i + p]$, for $1 \leq i \leq |w| - p$. A fundamental result dealing with periods is the following.

PROPOSITION 3.1 (FINE AND WILF [LOTHAIRE 1983]). *Let p, q be two periods of a word w. If $p + q \leq |w|$ then gcd(p, q) is also a period of w, where gcd stands for the greatest common divisor.*

We say that a word v *occurs* in a word w at position p if $w = w_1 v w_2$, for some words w_1, w_2 and $p = |w_1| + 1$. In this case, we say that p is a *starting position of an occurrence of the word v in w*. We say that an occurrence of v at position p in w *covers* a position k in w if $p \leq k \leq p + |v|$. We will use the following property of words:

LEMMA 3.2 (PLANDOWSKI 1999). *Let $i < j < k$ be three consecutive starting positions of occurrences of a word v in w. If $i + |v| \geq k$ (i.e., each of the occurrences of v covers the position k in w) then $k - j = j - i$ and $j - i$ is a period of the word $w[i, k + |v| - 1]$.*

PROOF. Since $i + |v| \geq k$, the occurrences of v at positions i and j overlap. Hence, $j - i$ is a period of v. Similarly, $k - j$ is a period of v. By $i + |v| \geq k$ we have $(k - j) + (j - i) = k - i \leq |v|$. By Proposition 3.1, $gcd(j - i, k - j)$ is also a period of v. Since there is no occurrence of v between positions i and j, we have $j - i = gcd(j - i, k - j)$. Similarly, we get $k - j = gcd(j - i, k - j)$. Hence, $j - i = k - j$. Let $p = k - j$. Then, since v occur at positions i and j in w, we have $w[s] = w[s + p]$, for $i \leq s < j$. Similarly, since v occurs at position j and k in w, we have $w[s] = w[s + p]$, for $j \leq s \leq k + |v| - 1 - p$. Hence, $p = k - j$ is a period of $w[i, k + |v| - 1]$. □

Let Σ and Ξ be two disjoint polynomial size alphabets: alphabet of constants and alphabet of variables. A *word equation* is a pair of words (u, v) (usually denoted by $u = v$) over the alphabet $\Sigma \cup \Xi$. A *length* of a word equation $e : u = v$ is defined by $|u| + |v|$ and denoted by $|e|$. A *solution* of the word equation is a morphism $h : (\Xi \cup \Sigma)^* \to \Sigma^*$ such that $h(a) = a$, for $a \in \Sigma$, and $h(u) = h(v)$. Note that a morphism being a solution of a word equation is uniquely determined by its values on variables. A solution h of $u = v$ is *minimal* if for all solutions g of $u = v$, $|h(u)| \leq |g(u)|$. There can be several minimal solutions of an equation.

An *index of periodicity* of a word w is a maximal number $p(w)$ such that $s^{p(w)}$, for a nonempty word s, is a subword of w. An *index of periodicity* $p(h, e)$ *of a solution* h of the equation e of the form $u = v$ is $p(h(u))$.

PROPOSITION 3.3 ([KOŚCIELSKI AND PACHOLSKI 1996]). *There is a constant c such that for each satisfiable equation e and its minimal solution h, $p(h, e) \leq 2^{c|e|}$.*

The original proof of Proposition 3.3 uses slightly different definition of a minimal solution and of an index of periodicity, but the same proof works also with our definitions.

Example 3.4. Let $\Sigma = \{a, b\}$ be a set of constants and $\Theta = \{X\}$ be a set of variables. Let e be a word equation $Xab = baX$. Then the set of all solutions $\{h_i\}$ of e is defined by $h_i(X) = (ba)^i b$. We have also $p(h_i, e) = p((ba)^{i+1}b) = i + 1$. h_0 is the only minimal solution of e.

A cardinality of a set S is denoted by $|S|$.

In our considerations, we deal with sequences of objects. The objects are letters of the alphabet (then the sequences are words) and index-factors that will be defined in Section 5.

Now we introduce some notation dealing with sequences. For a sequence $\mathcal{P} = s_1, \ldots, s_k$, the number k is the *length* of $\mathcal{P}$ and is denoted by length($\mathcal{P}$). Denote by last($\mathcal{P}$) the last element of $\mathcal{P}$ and by cut_last($\mathcal{P}$) the sequence $\mathcal{P}$ without last element of $\mathcal{P}$. For a positive integer t denote by $(\mathcal{P})^t$ a sequence consisting of t repetitions of $\mathcal{P}$. An *exponential expression* is an expression on sequences in which the only allowed operations are repetition and catenation of sequences. An exponential expression represents a sequence. The *size* of an exponential expression is its denotational length. The size of an expression can be defined recursively as follows.

$$\begin{aligned} \text{size}(\mathcal{P}, \mathcal{R}) &= \text{size}(\mathcal{P}) + \text{size}(\mathcal{R}) + 1 \\ \text{size}((\mathcal{P})^t) &= \text{size}(\mathcal{P}) + 3 \\ \text{size}(a) &= 1, \end{aligned}$$

where $\mathcal{P}$, $\mathcal{R}$ are exponential expressions and a is an exponential expression representing one-element sequence a. In the first case we add one for the period. In the second case we add 3 because one is for the exponent t and two is for two parenthesis. We assume that the denotational length of an exponent and an element of the sequence are one. For instance, the size of the expression $(a, b, c)^{100}$, b is 10 although it represents a sequence of length 301.

A *height* of an exponential expression is defined recursively. For an element a we define height(a) = 0. For sequences $\mathcal{P}$, $\mathcal{R}$ and an exponent t we define

$$\begin{aligned} \text{height}(\mathcal{P}, \mathcal{R}) &= \max(\text{height}(\mathcal{P}), \text{height}(\mathcal{R})), \\ \text{height}((\mathcal{P})^t) &= \text{height}(\mathcal{P}) + 1. \end{aligned}$$

The expressions we consider are mostly of height at most one. When we write of the expressions of other height we say their height. We say that two exponential expressions are *isomorphic* if they represent the same sequcence. For instance, the expression a, a, a is isomorphic to the expression $(a)^3$. Denote by seq($\mathcal{P}$) the sequence represented by exponential expression $\mathcal{P}$.

4. *The Relation* $\rightarrow$

The relation $\rightarrow$ is defined on $O(n^3)$ size descriptions of equations. The description is a pair of exponential expressions on constants and variables, that is, the expressions represent ordinary words over $\Sigma \cup \Theta$. One exponential expression describes the left-hand side of the equation and the second one the right-hand side of the equation. The exponential expressions are of height at most one and no variable is under repetition operator. Moreover, the exponents in the expressions does not exceed 2^{cn}. Clearly, such descriptions can be stored in polynomial space.

We say that two descriptions of equations are *isomorphic* if the expressions for their left hand sides are isomorphic and the expressions for their right hand sides are isomorphic.

Let e_1 and e_2 be $O(n^3)$ size descriptions satisfying the above conditions. Then $e_1 \rightarrow e_2$ if and only if e_2 is isomorphic to an equation, which can be obtained from e_1 by performing the following three steps.

(1) For each constant a in Σ, define an exponential expression exp_a over Σ of size $O(n^3)$ with exponents at most 2^{cn}. Then replace each constant a of e_1 by exp_a obtaining a description e_1'. The exponential expressions in e_1' can be of height two.

(2) For each variable X occurring in e_1 define an exponential expression exp_X over Σ such that seq(exp_X) is a prefix of each sequence represented by any exponential expression occuring directly to the right of X in e_1'. Then each occurrence of X in e_1' extend by exp_X, that is, replace fragments $X exp_X$, by X in e_1'. In this way, you obtain a description e_1''. Note here that replacement can result in computing exponential expressions for seq(exp_X)$^{-1}$seq(exp), if exp occurs directly to the right of some occurrence of X in e_1'. This however can be done in polynomial space.

(3) For each variable X that does not occur in e_1 but occurs in e_2 define an exponential expression exp_X over Σ. Then in some places of e_1'' replace exp_X by X. Note here that seq(exp_X) can occur inside seq(exp) for some exponential expression exp in e_1'' so this replacement can result in computing a description for some prefix of seq(exp) and some suffix of seq(exp). This however can be done in polynomial space.

The application of Steps (1–3) to e_1 may result in creating an equation that contains exponential expressions of height two and may be of size bigger than $O(n^3)$, but the isomorphism of it with e_2 can be checked in polynomial space due to the following lemma:

LEMMA 4.1. *The isomorphism of two descriptions e_1 and e_2 of equations containing exponential expressions of arbitrary height with exponents at most singly exponential can be done in polynomial space.*

PROOF. Note that the lengths of both sides of e_1 and e_2 are at most singly exponential so index of any position of the equations can be stored in polynomial number of bits. The arithmetic operations on such numbers can be of course performed in polynomial space. Suppose we want to check isomorphism of e_1 and e_2. First, we check whether the lengths of corresponding sides of e_1 and e_2 are identical. If not, then the equations are not isomorphic. Now, for each position j

we search for jth symbols of e_1 and e_2 and if they are different, then again e_1 and e_2 are not isomorphic. Otherwise, e_1 and e_2 are isomorphic. □

A more general result can be concluded from the result in Plandowski [1994, 1996], namely that isomorphism of two equations containing exponential expressions of arbitrary height can be decided deterministically in polynomial time.

LEMMA 4.2. *If $e_1 \to e_2$ and e_1 is satisfiable, then e_2 is satisfiable, too.*

PROOF. Let h be a solution of e_1. Using it we define a solution g of e_2. The replacement of constants of e_1 by expressions defines a morphism σ such that $\sigma(a) = exp_a$, for each constant a of e_1. The solution g of e_2 is defined by

$$g(X) = \begin{cases} \sigma(h(X))\text{seq}(exp_X) & \text{if } X \text{ occurs in } e_1 \\ \text{seq}(exp_X) & \text{if } X \text{ occurs in } e_2 \text{ but not in } e_1. \end{cases}$$ □

As a consequence of Lemma 4.2, we have the following lemma, which is an easy implication in Lemma 7.4.

LEMMA 4.3. *If $(a = a) \to^* e$, then e is satisfiable.*

The rest of the article is devoted to a proof of difficult implication of Lemma 7.4.

Example 4.4. We use now our algorithm to prove that the equation $Xab = baX$ is satisfiable. The proof is unnecessarily long but it shows how our algorithm works. We start from the equation $a = a$. Next, we execute Step (1) so a is replaced by an exponential expression $c^{10}d^{30}b$ obtaining the equation $c^{10}d^{30}b = c^{10}d^{30}b$. Step (2) does nothing because there are no variables in the equation. Now, we execute Step (3) by replacing in two places c^5 by X obtaining the equation $Xc^5d^{30}b = cXc^4d^{30}b$. Thus, we have

$$a = a \to Xc^5d^{30}b = cXc^4d^{30}b.$$

Now, we again execute Step (1) by replacing c by c, d by c^2 and b by b. We obtain the equation $Xc^5(c^2)^{30}b = cXc^4(c^2)^{30}b$ with exponenital expressions of height 2. Now we run Step (2) by replacing Xc^{10} by X in each place the variable X occurs in current equation. In this way, we obtain the equation isomorphic to $Xc^{55}b = cXc^{54}b$. Step (3) is empty because no new variable can appear in the equation. We have thus

$$Xc^5d^{30}b = cXc^4d^{30}b \to Xc^{55}b = cXc^{54}b.$$

Again we execute Step (1) by replacing c by ba and b by b. We obtain the equation $X(ba)^{55}b = baX(ba)^{54}b$. We execute Step (2) by replacing $X(ba)^{45}b$ by X. Again, Step (3) is empty. In this way, we obtain now

$$Xc^{55}b = cXc^{54}b \to X(ba)^{10}b = baX(ba)^9b.$$

Finally, we execute Step (1), which replaces b by b and a by a, Step (2), which replaces $X(ba)^9b$ by X, and Step (3), which is empty, and obtain

$$X(ba)^{10}b = baX(ba)^9b \to Xab = baX.$$

5. *Index-Factorizations*

A *factorization of a word* w is a sequence $\mathcal{P}$ of nonempty words $w_1, w_2, \ldots, w_k$ such that $w = w_1 w_2 \cdots w_k$. The words w_i are called *factors* of $\mathcal{P}$. Factors that are the same words are not distiguishable. We would like to generalize the notion of factorization to be able to distinguish factors occurring in different contexts. This is why we introduce index-factorizations. An *index-factorization of a word* w with indices I is a sequence of pairs

$$(w_1, i_1), (w_2, i_2), \ldots, (w_k, i_k)$$

such that $w = w_1 w_2 \cdots w_k$, and, for $1 \leq s \leq k$, $i_s \in I$, and w_s is a nonempty word. The pair (w_s, i_s) is called *index-factor*. The *length* of this index-factor is $|w_s|$.

In our considerations an *index-factorization* $\mathcal{F}$ is a function that takes a word and returns an index-factorization of this word. Denote by $\mathcal{F}(w)[i, j]$ a sequence of words that is obtained from $\mathcal{F}(w)$ by cutting it at positions corresponding to i and j and removing the outer parts, that is, remaining the part corresponding to $w[i, j]$. More precisely, let $\mathcal{F}(w) = (w_1, j_1), \ldots, (w_k, j_k)$ and let $i_t = |w_1 w_2 \cdots w_{t-1}| + 1$ for $1 \leq t \leq k+1$ (i_t, for $t < k+1$, is a starting position of an occurrence of the word w_t in w). Let s, r be such that $i_r \leq i < i_{r+1}$ and $i_s \leq j < i_{s+1}$. If $s = r$ then $\mathcal{F}(w)[i, j] = (w[i, j], j_r)$. Otherwise,

$$\begin{aligned}\mathcal{F}(w)[i, j] = {} & (w', j_r), (w_{r+1}, j_{r+1}), (w_{r+2}, j_{r+2}), \\ & \ldots, (w_{s-1}, j_{s-1}), (w'', j_s),\end{aligned}$$

where w' is a suffix of w_r of length $i_{r+1} - i$ and w'' is a prefix of w_s of length $j - i_s + 1$. We say that the subsequence $(w_r, j_r), (w_{r+1}, j_{r+1}), \ldots, (w_s, j_s)$ of $\mathcal{F}(w)$ (which is usually different from $\mathcal{F}(w)[i, j]$) *covers* the interval $[i, j]$ in $\mathcal{F}(w)$. For instance, if $\mathcal{F}(abababaab) = (aba, 2), (ba, 1), (ab, 3)$, then $\mathcal{F}(abababaab)[2, 6] = (ba, 2), (ba, 1), (a, 3)$.

Let $\mathcal{D}$ be a set of words of the same length and let \$ be a special symbol that does not occur in $\mathcal{D}$.

Definition 5.1. *A $\mathcal{D}$-factorization* is an index-factorization with indices $\mathcal{D} \cup \{\$\}$ which is defined as follows. If no word of $\mathcal{D}$ occurs in a word w, then $\mathcal{D}(w) = (w, \$)$. Otherwise, let $i_1 < i_2 < \cdots < i_k$ be the set of all starting positions of occurrences of the words of $\mathcal{D}$ in w and let $u_1, \ldots, u_k$ be the words in $\mathcal{D}$ that occur at these positions. If $i_1 > 1$, then

$$\begin{aligned}\mathcal{D}(w) = {} & (w[1, i_1 - 1], u_1), \ldots, \\ & (w[i_{k-1}, i_k - 1], u_k), (w[i_k, |w|], \$).\end{aligned}$$

If $i_1 = 1$, then

$$\begin{aligned}\mathcal{D}(w) = {} & (w[1, i_2 - 1], u_1), \ldots, \\ & (w[i_{k-1}, i_k - 1], u_k), (w[i_k, |w|], \$).\end{aligned}$$

For instance,

$$\{ab, bb\}(aabbaaababb) = (a, ab), (a, bb), (bbaa, ab), (ab, ab), (a, bb), (bb, \$).$$

LEMMA 5.2. *Let $\mathcal{D}$ be a set of words of the same length t. Let $i < j < k$ be three consecutive occurrences of a word $v \in \mathcal{D}$ in a word w. Assume that $i + t \geq k$. Then $\mathcal{D}(w)[i, j-1] = \mathcal{D}(w)[j, k-1]$.*

PROOF. By Lemma 3.2, $k-j = j-i$ and $k-j$ is a period of $u = w[i, k+t-1]$. It is enough to prove that for $0 \le p < j-i$ the words of length t starting at positions $i+p$ and $j+p$ in w are identical. This is true since these two words are wholy contained in u and the distance between their occurrences in u is equal to $j-i$ which is a period of u. □

Denote by $|\mathcal{D}|$ the cardinality of the set $\mathcal{D}$.

LEMMA 5.3. *Let $\mathcal{D}$ be a set of words of the same length t. Let $i < k$ be occurrences of two words $v, u \in \mathcal{D}$ in a word w. Assume that $i + t \ge k$. Then $\mathcal{D}(w)[i, k-1]$ can be represented by an exponential expression of size $O(|\mathcal{D}|^2)$.*

PROOF. Denote $\mathcal{P} = \mathcal{D}(w)[i, k-1]$. Let $i_1 = i < i_2 < \cdots < i_s = k$ be all starting positions of occurrences of words of $\mathcal{D}$ in w such that $i \le i_r \le k$. Note that for three consecutive occurrences of the same word in these occurrences Lemma 5.2 becomes applicable. Scan occurrences from left to right searching for the earliest two consecutive occurrences of the same word in $\mathcal{D}$. If each word in $\mathcal{D}$ appears only once then $s = \text{length}(\mathcal{P}) \le |\mathcal{D}|$ and we are done. Otherwise, let $j_1 < j_2$ be first starting positions of occurrences of a word v. Let $j_3 < \cdots < j_p$ be the other occurrences of v. Then by Lemma 5.2,

$$\mathcal{D}(w)[j_1, j_2 - 1] = \mathcal{D}(w)[j_2, j_3 - 1] = \cdots = \mathcal{D}[j_{p-1}, j_p - 1].$$

Moreover, by the choice of v, in $w[1, j_2-1+t]$ there is no two occurrences of a same word in $\mathcal{D}$. Hence, $\mathcal{P} = (\mathcal{P}_1)^1, (\mathcal{P}'_1)^{p-1}, \mathcal{P}'$, where $\text{length}(\mathcal{P}_1), \text{length}(\mathcal{P}'_1) \le |\mathcal{D}|$ and $\mathcal{P}'$ is determined by j_p and occurrences starting to the right of j_p. Note that among them there are no occurrences of v. We continue scanning starting from the position to the right of j_p. In this way, we obtain

$$\begin{aligned}\mathcal{P} = {} & (\mathcal{P}_1)^1, (\mathcal{P}'_1)^{p_1}, (\mathcal{P}_2)^1, (\mathcal{P}'_2)^{t_2}, \\ & \ldots, (\mathcal{P}_{q-1})^1, (\mathcal{P}'_{q-1})^{p_{q-1}}, (\mathcal{P}_q)^1.\end{aligned}$$

It is enough to prove that $q \le |\mathcal{D}|$. After each phase of scanning the set of words from $\mathcal{D}$ which occurs at remaining positions loses at least one element. This justifies the inequality $q \le |\mathcal{D}|$. This completes the proof. □

The most important property of this lemma is that the size of the expression *does not depend* on t but only on the cardinality of $\mathcal{D}$.

Let $\mathcal{P}$ be a sequence of words $w_1, \ldots, w_k$. Denote $\text{concat}(\mathcal{P}) = w_1 \cdots w_k$. The word $\text{concat}(\mathcal{P})$ is the concatenation of all elements of $\mathcal{P}$.

LEMMA 5.4. *Let $\mathcal{D}$ be a set of words of the same length t. Then*

$$\mathcal{D}(w)[i, j] = \text{cut\_last}(\mathcal{D}(w[i, j])), \mathcal{R},$$

where $\mathcal{R}$ can be represented by an exponential expression of size $O(|\mathcal{D}|^2)$, and $\text{concat}(\mathcal{R}) = \text{last}(\mathcal{D}(w[i, j]))$.

PROOF. Let $i_1 < \cdots < i_k$ be the starting positions of occurrences of the words in $\mathcal{D}$ in the word w. Denote by $\mathcal{I}$ the sequence $i_1, \ldots, i_k$. Let i_r be the first number in $\mathcal{I}$ that is not smaller than i, and i_s the last number in $\mathcal{I}$ that is not greater than j. Then, the definition of the sequence $\mathcal{D}(w)[i, j]$ is based on the positions $i_r, \ldots, i_s$. Let $j_1, \ldots, j_l$ be the starting positions of occurrences of the words in $\mathcal{D}$ in the word $w[i, j]$. Then, the definition of the sequence $\mathcal{D}(w[i, j])$ is based on the positions

in the sequence j_s, for $1 \leq s \leq l$. By the definition of the sequence $\mathcal{D}(w)[i, j]$, we have $s - r + 1 \geq l$ and

$$j_s = i_r - i + 1,\ j_{s+1} = i_{r+1} - i + 1, \ldots, j_l = i_{r+l-1} - i + 1.$$

Moreover, i_{r+l} is the first position in the sequence $i_r, i_{r+1}, \ldots, i_s$ such that $i_{r+l} + t > j$, that is, occurrences of words in $\mathcal{D}$ at positions $i_{r+l}, \ldots, i_s$ cover the positon j in w. Hence, cut_last($\mathcal{D}(w[i, j])$) is a prefix of $\mathcal{D}(w)[i, j]$ so that

$$\mathcal{D}(w)[i, j] = \text{cut\_last}(\mathcal{D}(w[i, j])), \mathcal{R}$$

where $\mathcal{R}$ is based on occurences of words in $\mathcal{D}$ at positions $i_{r+l-1}, i_{r+l}, \ldots, i_s$ in the word w. We have $i_{r+l} + t > j$ so that Lemma 5.3 becomes applicable for $i = i_{r+l}$, $k = i_s$. Thus, $\mathcal{R}$ can be represented by an exponential expression of size $O(|\mathcal{D}|^2)$.

Since

$$w[i, j] = \text{concat}(\mathcal{D}(w)[i, j]) = \text{concat}(\mathcal{D}(w[i, j])$$

and

$$\mathcal{D}(w)[i, j] = \text{cut\_last}(\mathcal{D}(w[i, j]), \mathcal{R},$$

we have concat($\mathcal{R}$) = last($\mathcal{D}(w[i, j])$). This completes the proof. □

An important property of Lemma 5.4 is that the part of factorization $\mathcal{D}(w)$ covering the interval $[i, j]$ consists of three parts: first one consisting of at most one index-factor, second one, namely $\mathcal{D}(w[i, j])$ without border factors, depending only on the word $w[i, j]$ but not of the position of its occurrence in w and the third one which is well compressible.

6. *Index-Factorizations of Solutions*

Fix a word equation $e : u = v$ and its minimal solution h. Denote $n = |e|$. A border is defined for a graphical representation of a word being a sequence of symbols written along a straight line. A *border* in a word w is a space between two consecutive symbols of w or a space before or after the word. Each word w contains $|w| + 1$ borders. Let $u = u_1u_2$ for some words u_1, u_2. A border between $h(u_1)$ and $h(u_2)$ in $h(u)$ is called *left cut*. Similarly, let $v = v_1v_2$ for some words v_1, v_2. A border between $h(v_1)$ and $h(v_2)$ in $h(v) = h(u)$ is called *right cut*. A *cut* in $h(u)$ is a border being left cut or right cut. Note that the borders before $h(u)$ and after it are left and right cuts and therefore the total number of cuts is at most n. A *special* word of $h(u)$ is a subword w of $h(u)$ such that there is an occurrence of w in $h(u)$ whose starting symbol is directly to the right of a cut. Formally, w is a special word if $h(u)$ can be written in form pws where p and s are such that there is a cut between p and ws.

Example 6.1. Let $e : XabY = YbaX$ and consider its solution $h(X) = b^3$, $h(Y) = b^3ab^3$. Then, the word $h(XabY) = h(YbaX) = b^3ab^4ab^3$ has eight cuts which are marked by dots in the following:

$$\cdot b^3 \cdot a \cdot b \cdot b^2 \cdot b \cdot a \cdot b^3 \cdot .$$

The words *bbb*, *bab* and *abb* are the only special words of length 3. The word *bba* is the only length 3 subword of $h(XabY) = h(YbaX)$ that is not a special word.

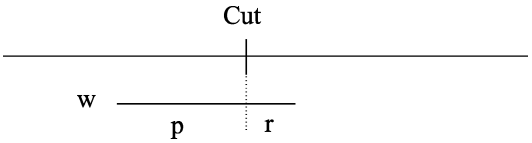

FIG. 1. w is divided by a cut into p and r.

Let $\mathcal{F}_l$ be the set of special words of length l. The set $\mathcal{F}_l$ is not empty for $1 \leq l \leq |h(u)|$. Clearly, for each l, $|\mathcal{F}_l| \leq n$. We present a new short proof of the following lemma.

PROPOSITION 6.2 (W. Rytter [Plandowski and Rytter 1998, Lemma 6]). *If h is minimal, then each subword of $h(u)$ of length at least 2 has an occurrence over a cut.*

PROOF. Suppose contrarily that there is a subword of $w = h(u) = h(v)$ that has no occurrence over a cut. Take the longest such subword p. First, we prove that occurrences of p inside $h(u)$ do not overlap. Indeed, suppose $p = w[i_1, j_1] = w[i_2, j_2]$ and $i_1 \leq i_2 \leq j_1$. Then, the word $w[i_1, j_2]$ has no occurrence over a cut. Otherwise, either its suffix p or its prefix p has an occurrence over a cut. Hence, p is not minimal. A contradiction. This means that the occurrences of p in w do not overlap. Let s be a word in which occurrences of the word p do not overlap and denote by $\Phi(s)$ the word that can be obtained from s by removing all occurrences of p. Define a morphism $h'(X) = \Phi(h(X))$ for each variable X of $u = v$. Since p does not have an occurrence over a cut, we have $h'(u) = \Phi(h(u)) = \Phi(w)$ and $h'(v) = \Phi(h(v)) = \Phi(w)$. Hence, $h'(u) = h'(v)$, that is, h' is a solution of $u = v$. Then, h is not minimal. A contradiction. □

Note here that, for $1 \leq l \leq |h(u)|$, the word $h(u) = h(v)$ starts from a special word of length l. Hence, for each index-factor (w, v) of $\mathcal{F}_l(h(u))$ of length at least l, a word in $\mathcal{F}_l$ is a prefix of w. In the following, we prove that index-factors in $\mathcal{F}_l(h(u))$ are short.

LEMMA 6.3. *Each index-factor in $\mathcal{F}_l(h(u))$ is of length at most $l(n+2)$.*

PROOF. The factorization $\mathcal{F}_l(h(u))$ is determined by occurrences of special words of length l in $h(u)$. Suppose there is an index-factor (w, v) in $\mathcal{F}_l(h(u))$ such that $|w| > l(n+2)$. By the definition of index-factorization $\mathcal{F}_l$, the prefix of w of length l is in $\mathcal{F}_l$ and there is no other occurrence of a word in $\mathcal{F}_l$ in w. By Proposition 6.2, the word w has an occurrence over a cut. The cut divides the word w into two parts p, r, (see Figure 1). We have $w = pr$. Since w does not contain an occurrence of word in $\mathcal{F}_l$ except as a prefix, we have $|r| < l$. Then $p > l(n+1)$ and again by Proposition 6.2, there is an occurrence of p (which is a prefix of w) over another cut. Then, $p = p'r'$ with $|p'| > l \cdot n$. We repeat the above with p' (which is a prefix of w) and so on up to the moment when we find a prefix of w which occurs over a cut which was hit earlier. Since there are at most n cuts this prefix say t is of length at least $2l$. The cut divides t into parts t' and t'' with $|t'| > l$, see (Figure 2). t' is contained in s—a prefix of w that hit the cut earlier in particular it is contained in w. It is impossible since t' as a prefix of w of length at least l starts as w with a word in $\mathcal{F}_l$ so that w contains this prefix. A contradiction, since w as a factor in $\mathcal{F}_l$-factorization can contain a word of $\mathcal{F}_l$ only as its prefix. □

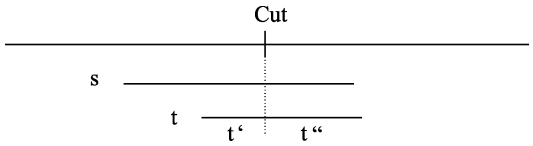

FIG. 2. t', a prefix of w of length at least l, is contained in s being a prefix of w.

7. *From Factorizations to Factor Equations*

As in the previous sections, we assume that e is an input equation of length n, h its minimal solution and $\mathcal{F}_l$ the set of special words of length l.

Let $\mathcal{D}$ be a set of words of the same length. A *factor equation over a set* $\mathcal{D}$ is a word equation in which an alphabet of constants is $\Sigma^* \times (\mathcal{D} \cup \{\$\})$ and an alphabet of variables is $\mathcal{G}$. Hence, constants are index-factors. The variables in $\mathcal{G}$ are called *factor variables*. A solution of a factor equation is a substitution of sequences of index-factors onto factor variables. We assume that each variable X of the word equation $e : u = v$ has a corresponding factor variable $\mathcal{X}$.

Consider the sequence of index-factors $\mathcal{P} = \mathcal{F}_l(h(u))$. The sequence $\mathcal{P}$ corresponds to a factor equation $\mathcal{E} : \mathcal{U} = \mathcal{V}$ over $\Sigma^* \times (\mathcal{F}_l \cup \{\$\})$ which we construct as follows: First, we construct $\mathcal{U}$ on the basis of u, h and $\mathcal{F}_l(h(u))$. Consider an occurrence of a variable X in u, that is, $u = u_1 X u_2$. Consider the occurrence of the word $h(X)$ in $h(u)$ at position $|h(u_1)|+1$. In the index-factorization $\mathcal{P}$ of $h(u)$, the interval $[|h(u_1)|+1, |h(u_1X)|]$ is covered by a sequence of index-factors (f_1, u_1), (f_2, u_2), $\ldots (f_k, u_k)$. By Lemma 5.4, applied with $\mathcal{D} = \mathcal{F}_l$, $w = h(u)$, $i = |h(u_1)| + 1$ and $j = |h(u_1X)|$ this sequence is of the form cut_last($\mathcal{F}_l(h(X))$), $\mathcal{R}$ where $\mathcal{R}$ can be described by an exponential expression of size $O(n^2)$. If cut_last($\mathcal{F}_l(h(X))$) $= \emptyset$ (the empty sequence), then we replace the subsequence (f_1, u_1), (f_2, u_2), $\ldots$, (f_k, u_k) in $\mathcal{P}$ by the expression $\mathcal{R}$. Otherwise, we replace it by the sequence: $\mathcal{X}$, $\mathcal{R}$ where $\mathcal{X}$ is the factor variable corresponding to X. Similarly, we treat all occurrences of variables in u. The sequence obtained from $\mathcal{P}$ in this way is $\mathcal{U}$. Similarly, we construct $\mathcal{V}$ on the basis of v, h and $\mathcal{F}_l(h(v)) = \mathcal{F}_l(h(u))$.

Observe, that in $\mathcal{E}$ there are only variables such that cut_last($\mathcal{F}_l(h(X))$) $\neq \emptyset$ and that the substitution $\mathcal{X} =$ cut_last($\mathcal{F}_l(h(X))$), for all X in e, which satisfy cut_last($\mathcal{F}_l(h(X))$) $\neq \emptyset$, is a solution of $\mathcal{E}$. Observe also that, although the length of $\mathcal{E}$ can be long, it can be represented by two exponential expressions (containing factor variables) of size $O(n^3)$. This does not mean, however, that the equation can be represented in PSPACE since the size of exponential expressions does not include the real size of index-factors and the size of exponents. The equation was constructed from an index-factorization of a minimal solution of an equation, so, by Proposition 3.3, each exponent in it can be encoded on cn bits. Index-factors are constants in $\mathcal{E}$, so the only important information is which of them are equal.

We say that a factor equation $\mathcal{E}$ is *equivalent* to an ordinary equation if they are identical up to renaming constants.

We have

LEMMA 7.1. *The factor equation $\mathcal{E}$ is equivalent to an ordinary equation that has description of size $O(n^3)$.*

Consider now two sequences $\mathcal{P}_1 = \mathcal{F}_{l+1}(h(u))$ and $\mathcal{P}_2 = \mathcal{F}_l(h(u))$. Since the set $\mathcal{F}_l$ contains all length l prefixes of the words in $\mathcal{F}_{l+1}$, the set of starting positions of

occurrences of words in $\mathcal{F}_{l+1}$ in $h(u)$ is a subset of the set of starting positions of occurrences of words in $\mathcal{F}_l$ in $h(u)$. Hence, the sequence $\mathcal{P}_2$ is obtainable from the sequence $\mathcal{P}_1$ by replacing its index-factors by sequences of index-factors that occur in $\mathcal{P}_2$. Moreover, since each index-factor (w, v) "remembers" the word $v \in \mathcal{F}_{l+1}$, which occurs just after w in $h(u)$, two identical index-factors in $\mathcal{P}_1$ are replaced by the same sequence of index-factors of $\mathcal{P}_2$. Moreover, as the following lemma says, the sequence can be represented by an exponential expression of size $O(n^3)$.

LEMMA 7.2. *Let the index-factor (w, v) be replaced by a sequence of index-factors $\mathcal{S} = (w_1, u_1), (w_2, u_2), \ldots, (w_k, u_k)$. Then the sequence $\mathcal{S}$ can be represented by an exponential expression of size $O(n^3)$.*

PROOF. Clearly, $w = w_1 w_2 \cdots w_k$. Moreover, $v = u_k a$, for some $a \in \Sigma$, or $v = u_k = \$$. Assume, that wv occurs at position i in $h(u)$. By Lemma 6.3, $|w| \leq (l+1)(n+2)$. We mark in the occurrence of w at i in $h(u)$ positions $i + l$, $i + 2l$, $i + 3l$, ... There are at most $\frac{(l+1)(n+2)}{l} + 1 \leq 2(n+2)$ marked positions. Since $|u_i| = l$, each occurrence of u_i covers at most one marked position. By Lemma 5.3, the sequence of index-factors $(w_i, u_i), \ldots, (w_j, u_j)$, for $i < j$, such that the occurrences of $u_i, \ldots, u_j$ in $h(u)$ cover the same marked position can be represented by an exponential expression of size $O(n^2)$. Since the number of marked positions is linear in n, the result follows. □

Denote by $\mathcal{E}_l$ the factor equation that corresponds to the sequence of index-factors $\mathcal{F}_l(h(u))$. We will show now how to transform $\mathcal{E}_{l+1}$ onto $\mathcal{E}_l$, nondeterministically. First, we replace constants of $\mathcal{E}_{l+1}$ by sequences of index factors as in Lemma 7.2. By Lemma 7.2, these sequences can be represented by exponential expressions of size $O(n^3)$. In this way, we obtain an equation $\mathcal{E}'$. The set of constants of $\mathcal{E}'$ is now the same as the set of constants of $\mathcal{E}_l$. Variables may occur in $\mathcal{E}_l$ that do not occur in $\mathcal{E}'$. Such a variable $\mathcal{X}$ has the property that a corresponding variable X in e satisfies cut_last($\mathcal{F}_{l+1}(h(X)) = \emptyset$ and cut_last($\mathcal{F}_l(h(X)) \neq \emptyset$. We replace sequences cut_last($\mathcal{F}_l(h(X))$ in $\mathcal{E}'$ that correspond to occurrences of $\mathcal{X}$ in $\mathcal{E}_l$ by $\mathcal{X}$ obtaining equation $\mathcal{E}''$. Now the sets of factor variables and constants of $\mathcal{E}_l$ to $\mathcal{E}''$ are the same. Now we take care of the other factor variables, i.e. such that their corresponding variables X satisfy cut_last($\mathcal{F}_{l+1}(h(X)) \neq \emptyset$ and cut_last($\mathcal{F}_l(h(X)) \neq \emptyset$. The word concat(cut_last($\mathcal{F}_{l+1}(h(X))$) can be a proper prefix of concat(cut_last($\mathcal{F}_l(h(X))$). In this case, the variable $\mathcal{X}$ in $\mathcal{E}''$ expands to the right, that is, for some sequence $\mathcal{R}$ to the right of each occurrence of $\mathcal{X}$ in $\mathcal{E}''$ there is a sequence which starts from $\mathcal{R}$. The occurrences of $\mathcal{X}$,$\mathcal{R}$ are thus replaced by $\mathcal{X}$. In this way we obtain the equation $\mathcal{E}_l$.

Observe also that transformation from $\mathcal{E}_{l+1}$ to $\mathcal{E}_l$ follows the transformation that is used to define the relation $\rightarrow$. We cannot write $\mathcal{E}_{l+1} \rightarrow \mathcal{E}_l$ because the constants of $\mathcal{E}_{l+1}$ and $\mathcal{E}_l$ are of unbounded size and the lengths of $\mathcal{E}_{l+1}$ and $\mathcal{E}_l$ can be too large. We can however claim that there are two descriptions e_{l+1} and e_l of ordinary equations such that $e_{l+1} \rightarrow e_l$ and e_{l+1} is a decription of an ordinary equations equivalent to $\mathcal{E}_{l+1}$ and e_l is a description of an ordinary equation equivalent to $\mathcal{E}_l$.

Observe that the equation e is obtainable from the descrription e_1 in the same way as e_l from e_{l+1}, that is, $e_1 \rightarrow e$.

Example 7.3. Denote $w_i = (ba)^i b$, $u_i = (ab)^i a$. Consider equation $e : Xab = baX$ and its solution g defined by $g(X) = w_{99}$. Although g is not minimal the

factorizations $\mathcal{F}_l$ and equations $\mathcal{E}_l$ are well defined for all solutions. The only special word of length 201 is w_{100} and the only special words of length $2i+1$, for $0 \leq i \leq 99$, are w_i and u_i. Then

$$\mathcal{F}_{2i+1}(h(Xab)) = [(b, u_i)(a, w_i)]^{100-i}(w_i, \$)$$

and $\mathcal{E}_{201} : (w_{100}, \$) = (w_{100}, \$)$, and

$$\mathcal{E}_{199} : (b, u_{99})(a, w_{99})(w_{99}, \$) = (b, u_{99})(a, w_{99})(w_{99}, \$),$$

and, for $0 \leq i \leq 98$,

$$\mathcal{E}_{2i+1} : \mathcal{X}(b, u_i)(a, w_i)(w_i, \$) = (b, u_i)(a, w_i)\mathcal{X}(w_i, \$).$$

Note that, for $0 \leq i \leq 98$, the equations $\mathcal{E}_i$ identical up to renaming constants.

Now we are ready to prove Lemma 7.4.

LEMMA 7.4. *$(a = a) \rightarrow^* e$ if and only if e is satisfiable.*

PROOF. If $e : u = v$ is solvable, then we take its minimal solution h. We construct factor equations $\mathcal{E}_l$ corresponding to sequences $\mathcal{F}_l(h(u))$. There is a description $e_{|h(u)|}$ such that $e_{|h(u)|} \rightarrow^* e$ and $\mathcal{E}_{|h(u)|}$ is equivalent to $e_{|h(u)|}$. Since $\mathcal{E}_{|h(u)|}$ is equivalent to $a = a$, the result follows.

If $(a = a) \rightarrow^* e$, then the result follows from Lemma 4.2. □

THEOREM 7.5. *The problem of satisfiability of word equations is in PSPACE.*

ACKNOWLEDGMENTS. I would like to thank Prof. Wojciech Rytter (Warsaw University, Poland) for introducing me to the field of word equations and for many useful discussions.

I would like to thank Prof. Filippo Mignosi (University of Palermo, Italy) for many discussions during my visit in Palermo in 1997. The discussions were about synchronizing factorizations introduced in Karhumaeki et al. [2000]. I used in this paper two of his ideas. The first one was to use several words instead of one in the definition of $\mathcal{F}_Q$-factorization, see Karhumaeki et al. [2000]. The second idea was to use context in the definition of a synchronizing factorization. Both ideas are utilized in the definition of $\mathcal{D}$-factorization which, although is not an example of synchronizing factorization, is very close to it.

I would like to thank an anonymous referee whose detailed comments improved the final presentation of the article.

REFERENCES

ABDULRAB, H. 1987. Resolution d'equations sur le mots: Etude et implementation LISP de l'algorithme de Makanin. Ph.D. dissertation, Universite de Rouen, Rouen, France.

ABDULRAB, H., AND PÉCUCHET, J. 1989. Solving word equations. *J. Symb. Comput. 8*, 499–521.

ANGLUIN, D. 1979. Finding pattern common to a set of string. In *Proceedings of Symposium on the Theory of Computing (STOC'79)*. ACM, New York, 130–141.

BAADER, F., AND SNYDER, W. 2001. Unification theory. In *Handbook of Automated Reasoning Vol. 1*, J. Robinson and A. Voronkov, Eds. Elsevier Science Publishers, 447–533.

DIEKERT, V. 2002. Makanin's algorithm. In *Algebraic Aspects of Combinatorics on Words*, M. Lothaire, Ed. Cambridge University Press.

GUTIERREZ, C. 1998. Satisfiability of word equations with constants is in exponential space. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS'98)*. IEEE Computer Society Press, Los Alamitos, Calif, 112–119.

JAFFAR, J. 1990. Minimal and complete word unification. *J. ACM 37*, 1, 47–85.

KARHUMAEKI, J., MIGNOSI, F., AND PLANDOWSKI, W. 2000. The expressibility of languages and relations by word equations. *J. ACM 47,* 5, 483–505.

KOŚCIELSKI, A., AND PACHOLSKI, L. 1996. Complexity of Makanin's algorithm. *J. ACM 43,* 4, 670–684.

LOTHAIRE, M. 1983. *Combinatorics on Words*. *Encyclopedia of Mathematics and Its Applications*, Vol. 17. Addison-Wesley, Reading, Mass. (Reprinted in the *Cambridge Mathematical Library*, Cambridge University Press, 1997.)

MAKANIN, G. 1977. The problem of solvability of equations in a free semigroup. *Matematiceskij Sbornik 103,* 2, 147–236. (In Russian; English translation in: *Math. USSR Sbornik 32,* 129–198, 1977.)

PÉCUCHET, J. 1981. Equations avec constantes et algorithme de Makanin. Ph.D. dissertation, Laboratoire d'informatique in Rouen, Rouen, France.

PLANDOWSKI, W. 1994. Testing equivalence of morphisms on context-free languages. In *Proceedings of the European Symposium of Algorithms (ESA'94)*, Lecture Notes in Computer Science, Vol. 855. Springer-Verlag, New York, 460–470.

PLANDOWSKI, W. 1996. The complexity of the morphism equivalence problem for context-free languages. Ph.D. dissertation, Warsaw University, Warsaw, Poland.

PLANDOWSKI, W. 1999. Satisfiability of word equations is in NEXPTIME. In *Proceedings of the Symposium on the Theory of Computing (STOC'99)*. ACM, New York, 721–725.

PLANDOWSKI, W., AND RYTTER, W. 1998. Application of lempel-ziv encodings to the solution of word equations. In *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'98)*, Lecture Notes in Computer Science, Vol. 1443. Springer-Verlag, New York, 731–742.

SCHAEFER, M., SEDGWICK, E., AND STEFANKOVIC, D. 2002. Recognizing string graphs is in np. In *Proceedings of the Symposium on the Theory of Computing (STOC'02)*. ACM, New York, 1–6.

SCHULZ, K. 1992. Makanin's algorithm for word equations: Two improvements and a generalization. In *Proceedings of the International Workshop on Word Equations and Related Topics (IWWERT'90)*, Lecture Notes in Computer Science, Vol. 572. Springer-Verlag, New York, 85–150.

RECEIVED OCTOBER 2002; REVISED AUGUST 2003; ACCEPTED OCTOBER 2003